microprocessori

dai chips ai sistemi

EDIZIONE
ITALIANA

RODNAY ZAKS

GRUPPO
EDITORIALE
JACKSON

microprocessori

microprocessori

microprocessori

dai chips ai sistemi

di
Rodnay Zaks

GRUPPO
EDITORIALE
JACKSON
Via Rossellini, 12
20124 Milano

Hanno contribuito:
Disegni tecnici: Carl Bonner. **Copertina:** Daniel Lenoury. **Coordinazione:** Barry Janoff, Ruth Berling, Carl Bonner. **Composizione:** Linda Hodges, Linda Richman.

AVVERTENZE

Si è cercato per quanto possibile, di fornire informazioni complete e rigorose. In ogni caso la Sybex non si assume alcuna responsabilità per il loro impiego; nemmeno al riguardo di infrazioni di brevetti e di altri diritti di terze parti che ne potrebbero derivare. I costruttori di apparecchiature non rilasciano alcuna autorizzazione su apparecchiature protette da brevetto o diritti di brevetto e si riservano la facoltà di cambiare, in qualunque momento, la disposizione circuitale senza alcun preavviso.

In particolare sono soggetti a frequente cambiamento le caratteristiche tecniche e i prezzi. I confronti e le valutazioni sono presentati solo per il loro valore educativo ed i loro principi informativi. Per le specifiche esatte si rimanda il lettore ai dati del costruttore.

Stampato in Italia da
S.p.A. Alberto Matarelli - Milano Stabilimento Grafico

PREFAZIONE

PER CHI?

Per leggere questo libro non è richiesta alcuna conoscenza preliminare dei calcolatori o dei microprocessori, sebbene una conoscenza tecnica di base sia naturalmente un vantaggio. I microprocessori ed altri chips LSI hanno reso il progetto di un sistema così semplice che non è richiesta alcuna significativa preparazione scientifica o elettronica.

Questo libro si rivolge perciò a tutti coloro che hanno solamente una conoscenza frammentaria o addirittura nessuna conoscenza riguardo al mondo dei microprocessori e che desiderino comprendere tutti i concetti, le tecniche e i componenti in breve tempo. Può essere utilizzato da non specialisti: studenti, hobbysti come anche da ingegneri. La ragione sta nella sua completezza e progressività.

UNA BREVE STORIA DI QUESTO LIBRO

L'autore si è dedicato al progetto di microprocessori per uso industriale nella Silicon Valley fin dal 1972, così come al progetto generale di calcolatori prima di quella data. Inoltre ha insegnato progetto con microprocessori a piú di 2000 persone, sia studenti che ingegneri esperti di calcolo numerico. Questo libro è il risultato delle esperienze di progetto e di insegnamento dell'autore. Rappresenta la settima versione di un corso precedente e di libri preparati per seminari e pubblicazioni scritte dall'autore negli anni scorsi. In particolare include il testo dei seminari C10 e A1 della SYBEX.

Come risultato di questo esteso «testing e debugging» questo libro dovrebbe essere completamente autosufficiente. La struttura di ciascun capitolo è progressiva: dalle definizioni fondamentali alle applicazioni avanzate — dai principi alle tecniche, agli esempi applicativi.

CIÒ CHE IMPARERETE

Il Capitolo 1 vi introdurrà a tutti i concetti e le definizioni di base.

Il Capitolo 2 vi mostrerà, in dettaglio, come opera un reale microprocessore.

Il Capitolo 3 vi presenterà le altre tecniche e i componenti richiesti per sviluppare le funzioni di memoria e di input/output.

Il Capitolo 4 discuterà i meriti relativi di ciascuno dei microprocessori principali.

Il Capitolo 5 vi indicherà come collegare tutti i componenti precedenti in un sistema.

Il Capitolo 6 presenta alcune applicazioni: come realizzarle, quali siano le differenze.

Il Capitolo 7 è un capitolo piú impegnativo: come interfacciare il nostro sistema base alle periferiche tipiche (compreso il Bus S-100).

Il Capitolo 8 presenta i principi fondamentali e le tecniche di programmazione.

Il Capitolo 9 vi mostrerà i problemi rimanenti e gli strumenti disponibili per sviluppare un sistema reale.

Il Capitolo 10 presenterà predizioni riguardanti l'evoluzione futura.

COME LEGGERE IL LIBRO

Una lettura in sequenza è raccomandata, ma non è necessaria. Se non siete interessati o preparati per l'interfacciamento potete tralasciare il Capitolo 7. In egual modo se il funzionamento interno del microprocessore non vi interessa, potete tralasciare il Capitolo 2.

Poichè ogni capitolo ed ogni paragrafo sono fortemente strutturati dall'argomento semplice al complesso, è possibile indirizzarsi alle varie sezioni con un ordine qualsiasi. Tuttavia è fortemente raccomandabile che l'intero libro venga letto interamente almeno una volta. Ciascun capitolo è stato congegnato per insegnarvi un aspetto specifico. I capitoli successivi si sviluppano a partire dalla conoscenza presentata nei precedenti. Diversi esercizi non hanno alcuna risposta. Dovreste essere capaci di rispondere alle domande e sentirvi sicuri di possedere la risposta esatta.

PER IMPARARE RAPIDAMENTE COME FUNZIONA UN SISTEMA

Se il vostro tempo è molto limitato, leggete i Capitoli 1-2-3 e 5, senza neppure scorrere le pagine sui componenti reali. In poche ore saprete come funziona un sistema e come è costruito.

ALTRI AUSILI DIDATTICI

Sono disponibili dalla SYBEX corsi programmati su cassette nonchè altri libri e seminari in sede.

MICROPROCESSORI

L'industria dei microprocessori è probabilmente quella a crescita piú rapida oggi. I suoi effetti sull'economia sono stati qualificati come «seconda rivoluzione industriale». Un microcomputer completo può essere sviluppato già ora su un singolo chip per un dollaro in quantitativi di grande serie. Esso rende disponibile, per tutti i processi che ci circondano, una «intelligenza programmata».

La semplicità d'uso è tale che rimangono solo due limitazioni: l'ingenuità dell'utilizzatore e la sua competenza. Questo libro si indirizza a rimuovere la seconda limitazione.

SOMMARIO

CAPITOLO 1

CONCETTI FONDAMENTALI

I concetti fondamentali e le definizioni necessarie per comprendere i microprocessori e i microcomputer saranno introdotte in questo capitolo.

Il *microprocessor* è un nuovo componente LSI che realizza la maggior parte delle funzioni di un processore tradizionale su di un unico chip. Nuovi termini verranno definiti progressivamente attraverso tutto questo capitolo.

LSI sta a significare «Large Scale Integration» (n.d.t. cioè intergrazione su larga scala di moltissime funzioni in contrasto all'integrazione su media scala MSI e su piccola scala SSI) e si riferisce alla nuova tecnologia nella quale diverse migliaia di transistori possono essere integrate su un unico circuito integrato (CI).

Un *chip* è un pezzetto rettangolare di silicio sul quale questo circuito integrato viene realizzato. Il processo di fabbricazione in LSI sarà descritto più avanti in questo capitolo.

Un *microcomputer* è un computer la cui unità centrale di elaborazione «Central Processing Unit» (CPU) è stata realizzata usando un microprocessore LSI. Il progresso della tecnologia LSI permette già la realizzazione di un semplice ma completo computer su di un unico chip («microcomputer - on - a chip»).

Il primo microprocessore fu introdotto sul mercato alla fine del 1971 (I4004) ed ora sul mercato esistono più di 30 tipi diversi di microprocessori. Solamente alcuni, tuttavia, sono venduti in quantità significativa. Il tipico prezzo di vendita al momento della stesura di questo testo è di $10,00 in quantitativi di 100 o più pezzi (un tale prezzo ha valore unicamente indicativo e può variare rapidamente con le condizioni di mercato).

Un *sistema con microprocessore* è un sistema completo che include funzioni di elaborazione, memoria e dispositivi di ingresso e di uscita. Il microcomputer è naturalmente un sistema con microprocessore. Tuttavia generalmente si pensa ad un microcomputer come ad un computer da tavolo, cioè come ad un sistema in un contenitore con pannello frontale ed alimentatore adatto ad essere usato come computer di impiego generale.

PRINCIPI DI FUNZIONAMENTO

I principi di funzionamento dei sistemi a microprocessore sono naturalmente identici a quelli di qualsiasi altro computer. Tuttavia, i nuovi sistemi sviluppati uti

lizzando microprocessori differiscono spesso significativamente dai sistemi tradizionali. La ragione fondamentale è che *il processore di per se stesso è diventato una delle risorse* meno costose del sistema. In tali condizioni l'elemento di elaborazione può essere considerato concettualmente come periferica ad altri componenti del sistema, in particolare le unità di input/output. In breve ci si può aspettare per il prossimo futuro che la maggior parte dei cosiddetti chips con funzioni periferiche disponibili oggi per l'impiego su sistemi a microprocessori includeranno un *modulo processore* con funzioni fondamentali. I chips periferici stanno diventando «muniti di processore» e perciò programmabili.

I concetti fondamentali di funzionamento di un sistema calcolatore sono passati in rassegna qui di seguito. Nel Capitolo 3 verranno invece passati in rassegna ed esaminati nel dettaglio i nuovi componenti LSI disponibili per costruire un sistema a microprocessore. Infine nel Capitolo 5 questi componenti verranno interconnessi fino a formare un completo sistema funzionale.

La struttura fondamentale di un sistema calcolatore è illustrata in Fig. 1-1. Un sistema include 6 unità fondamentali. Al centro dell'illustrazione compare l'*Unità Centrale di Processo,* in inglese Central Processing Unit, abbreviata in CPU. La CPU include 2 unità: l'*Unità di Controllo* (CU) e l'*Unità Logica-Aritmetica* (ALU).

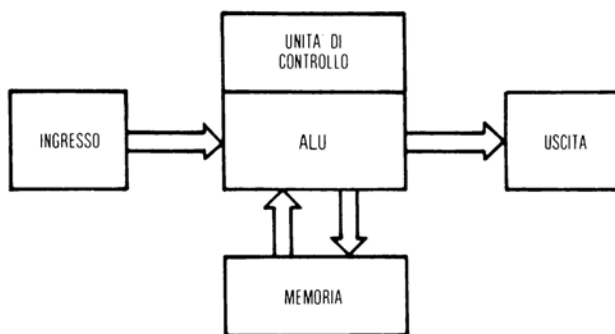



Fig. 1-1: Un sistema a computer è formato da 5 unità funzionali.

La funzione dell'*Unità Logica-Aritmetica* è di eseguire le operazioni aritmetiche e logiche sui dati che l'attraversano. Le tipiche funzioni aritmetiche includono l'addizione e la sottrazione. Le tipiche operazioni logiche includono le operazioni di «and logico», di «or logico» e di shift.

La responsabilità dell'*Unità di Controllo* è di dare una sequenza alle operazioni dell'intero sistema. In particolare essa genererà e coordinerà tutti i segnali di con-

trollo necessari per sincronizzare le operazioni e il flusso dei dati entro l'Unità Logica-Aritmetica così come al di fuori di essa. Essa controllerà il flusso dei dati nell'address-bus (bus degli indirizzi) e nel data-bus (o bus dei dati) e coordinerà o interpreterà i segnali presentati sul control-bus (o bus di controllo) del sistema. (Un *bus* è un insieme di segnali o linee raggruppate secondo la funzione — i tre bus standard in un sistema a microprocessore sono il data bus, l'address bus e il control bus). Uno dei ruoli essenziali dell'unità di controllo è di *reperire* (fetch), *decodificare* ed *eseguire* istruzioni successive immagazzinate nel sistema di memoria. Una tale sequenza di istruzioni è chiamata *programma.* La CU è generalmente associata fisicamente all'ALU che essa controlla e la combinazione della CU ed ALU è chiamata *Central Processing Unit* o CPU. Un microprocessore è fondamentalmente una CPU su un singolo chip.

La CPU non viene necessariamente sviluppata come un singolo componente e la CU può essere separata dall'ALU. In particolare i *bit-slices* ora disponibili realizzano la sezione ALU di un calcolatore tradizionale separatamente dalla sezione di controllo, che deve essere progettata e montata a parte. L'architettura bit-slice sarà esaminata al Capitolo 5.

Tornando alla Fig. 1-1, il *modulo di memoria* appare al di sotto della CPU, in fondo all'illustrazione. La memoria di qualsiasi sistema calcolatore è usata per memorizzare l'*informazione.* Si possono distinguere fisicamente diversi tipi di memorie in ragione del fatto che si possa scrivere l'informazione nella memoria e poi leggerla oppure che si possa solamente leggere da quest'ultima. Questi due tipi fondamentali di memorie sono chiamati rispettivamente *RAM* (Random Access Memory che significa Memoria ad accesso casuale) e *ROM* (Read Only Memory che significa Memoria a sola lettura).

Una *RAM* è una memoria nella quale i dati possono essere sia scritti che letti. Il tipico tempo di ciclo di una tale memoria varia da 500 nanosecondi (ns) ad 1 microsecondo (μs) per le RAM MOS LSI. La *ROM* è una memoria che può essere solamente letta una volta che i dati sono stati depositati in essa. Il suo principale vantaggio è di essere *non volatile* mentre le RAM standard sono volatili, cioè perdono i loro dati quando manca alimentazione. Funzionalmente la memoria contiene due tipi di informazione: programmi e dati.

Un programma è una sequenza di istruzioni che è stata scritta dall'utilizzatore, quindi codificata nel sistema binario, cosicchè possa risiedere in una memoria elettronica. Ciascuna delle *istruzioni* in sequenza di un programma verrà reperita (fetched) sotto la supervisione dell'unità di controllo e depositata in un registro speciale di questa unità di controllo, dove sarà decodificata ed eseguita. Ad esempio una tipica istruzione potrebbe addizionare tra loro i contenuti di due registri e depositare i risultati in un terzo registro. Questo procedimento verrà studiato al Capitolo 2.

I *dati* contenuti nella memoria verranno elaborati dall'Unità Logica-Aritmetica. I dati possono avere diversi formati. I più importanti saranno descritti al Capitolo

8. Tipicamente i dati sono numeri o caratteri, rappresentati nel sistema binario.

La *struttura* di una memoria verrà descritta nel paragrafo successivo. Le *caratteristiche tecniche* delle memorie LSI verranno esaminate al Capitolo 3.

Ritornando alla Fig. 1-1, i due moduli rimanenti sono rispettivamente il *modulo di input* e il *modulo di output.*

Il *modulo di input* appare sulla sinistra dell'illustrazione e fornisce i dati all'ALU. Esso può essere, ad esempio, una tastiera o un sensore (sensore di temperatura rivelatore di presenza, sensore di pressione). In un tipico microcomputer il dispositivo usuale di ingresso è una tastiera.

Il *modulo di output* renderà disponibili al mondo esterno i dati provenienti dall'ALU. Tipici moduli di output sono LED's (Light-Emitting Diodes che significa Diodi emettitori di luce) o LCD's (Liquid-Crystal-Displays che significa Visualizzatori a cristalli liquidi), che sono estensivamente utilizzati sugli orologi digitali e le calcolatrici da tasca, stampanti, lampadine o qualsiasi meccanismo di controllo (come un motore o un relé, per esempio).

I moduli di ingresso e di uscita non sono normalmente connessi direttamente all'ALU ma ai *bus* del sistema, cioè agli address-bus, data-bus, control-bus. Per questa ragione, i dati provenienti dal modulo di ingresso non *esigono* necessariamente di transitare attraverso l'ALU per raggiungere la memoria o eventualmente il modulo di uscita. Tuttavia nell'architettura della maggior parte dei sistemi e in particolare dei sistemi a microprocessore, questo è proprio ciò che avviene. Tutti i dati in un tipico sistema a microprocessore generalmente transitano attraverso un registro speciale dell'ALU chiamato *accumulatore.* Tuttavia esiste la possibilità di realizzare altri schemi di trasmissione. Ciò verrà usato in particolare nel caso del DMA (Direct Memory Access Controller che significa Controllore ad accesso diretto in memoria), che verrà studiato al Capitolo 3.

I BUS

Un bus è stato definito come un sistema di segnali raggruppati per funzione. Gli elementi di un sistema sono interconnessi per mezzo di tre bus: il data-bus, l'address-bus e i control-bus.

Il *data-bus* trasmette i *dati* tra le unità. Un microprocessore a otto bit richiede un data bus di otto bit al fine di trasmettere otto bit di dato in parallelo. Il data-bus è bidirezionale (trasmette in entrambe le direzioni).

L'*address-bus* è utilizzato per *selezionare* l'origine o la destinazione dei segnali trasmessi su uno degli altri bus. Tipicamente esso è usato per selezionare un registro all'interno di una delle unità del sistema che verrà utilizzato come sorgente o destinazione dei dati. Tradizionalmente un address-bus standard ha 16 linee cioè può indirizzare $2^{16} = 64K$ dispositivi ($1K = 1024$ nel gergo dei calcolatori).

Il *control-bus* è usato per la *sincronizzazione* del sistema. Esso contiene informazioni di stato e di controllo sia dalla che verso l'unità microprocessore (l'abbreviazione standard per l'unità microprocessore è ora diventata MPU). Un control—bus minimale richiede almeno 10 o preferibilmente piú linee di controllo per svolgere la propria funzione.

ESEMPIO: UNA CALCOLATRICE TASCABILE

Come esempio di un semplice sistema di calcolo esaminiamo il funzionamento di una calcolatrice tascabile. I moduli che appaiono in Fig. 1-2 sono stati raggruppati in modo differente che nella precedente illustrazione. Il *microprocessore* costituisce la CPU cioè le funzioni di controllo e le funzioni di calcolo. La *memoria* immagazzina il programma e i dati. Il modulo di *input/output* è l'interfaccia speciale richiesta per collegare il modulo di ingresso e il modulo di uscita al microprocessore. Il modulo di ingresso qui è una tastiera esadecimale cioè con 16 tasti. Il modulo di uscita è un visualizzatore LED.

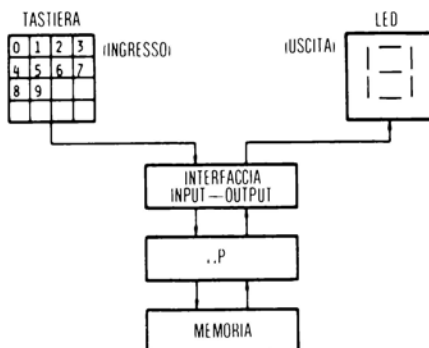



Fig. 1-2: Un calcolatore tascabile è un microcomputer semplificato.

Una sequenza tipica è la seguente:

1. L'utente imposta un numero sulla tastiera.

2. Questo numero verrà trasferito in un registro del sistema contenuto sul microprocessore. Assumiamo qui che quello trattato sia un modello standard contenente due registri interni. Ciascun registro può immagazzinare un numero decimale di 8 cifre, piú il segno. Questi due registri sono sufficienti nel caso di semplici operazioni, come «+» o «—» e nessun dato ha bisogno di essere depositato nella memoria. A questo punto la memoria sarebbe necessaria solo per conservare il *programma*. Infatti questa memoria non è necessariamente *esterna* al microprocessore stesso. Per chiarezza nella figura la memoria appare distinta dal microprocessore vero e proprio.

3. Una volta che il numero decimale completo sia stato trasferito in un registro (fino ad 8 cifre decimali), dalla tastiera verrà specificata una operazione. Questa operazione specificata sarà memorizzata in uno speciale registro del microprocessore fino a quando potrà essere eseguita. Supponiamo che sia stato specificato un «+».

4. Il secondo operando richiesto dall'operazione specificata sarà poi impostato dall'utente sulla tastiera. Ancora un completo numero decimale viene accumulato fino a che l'utente preme un altro tasto operativo come ad esempio «=».

5. La specificazione dell'operazione «=» è equivalente ad ordinare l'esecuzione di un *programma aritmetico* determinato dall'*operatore* che è stato preservato in uno speciale registro («+» nel nostro caso). Allora viene eseguito un programma di addizione: le istruzioni di quel programma depositate nella memoria vengono eseguite. Come risultato viene eseguita l'addizione e la somma viene depositata in uno dei registri inizialmente riservati agli operandi che viene chiamato *accumulatore*. Questo risultato può ora essere visualizzato per l'utente.

6. Il risultato viene inviato dall'accumulatore (contenuto nel microprocessore) all'unità di visualizzazione LED, dove è visualizzato per l'utente. Questo risultato rimane memorizzato nell'accumulatore interno fino a che non viene esplicitamente cancellato utilizzando il tasto «C» (iniziale di *clear*) o altrimenti fino a che l'utente non specifica una nuova operazione.

Questa semplice sequenza illustra l'uso dei moduli principali di un sistema di calcolo: i dispositivi di ingresso e uscita, la memoria (che qui aveva solamente la funzione di memorizzare un programma) e la Central-Processing-Unit che esegue le operazioni e stabilisce le sequenze del sistema. La maggior parte delle CPU incorpora una piccola quantità di memoria realizzata in registri interni veloci onde migliorare l'efficienza. Nell'esempio precedente l'unità di elaborazione incorporava almeno due registri capaci di memorizzare l'operazione specificata. Concettualmente questi registri possono essere considerati come parte della memoria del sistema. Diversi «livelli» di memoria sono disponibili in uno stesso sistema. I registri sono i livelli di memoria piú veloci.

Le tecniche di lettura delle informazioni da una tastiera o di trasformazione di un numero decimale in un codice adatto ai visualizzatori LED verranno studiate in dettaglio al Capitolo 7 (Tecniche di Interfacciamento).

LA MEMORIA

La memoria del sistema è utilizzata per immagazzinare sia i programmi che verranno eseguiti sul processore, sia i dati che verranno manipolati dal sistema. La memoria può essere realizzata in almeno tre maniere:

1. **I registri interni** che fanno parte di solito del modulo ALU provvedono il livello più veloce per la memorizzazione di dati disponibile nel sistema. Tipicamente i con-

tenuti dei registri interni possono essere utilizzati dal sistema in meno di 100 ns (1 nanosecondo = 10^{-6} s).

2. **La memoria principale** del sistema è generalmente realizzata con uno o diversi componenti e la dimensione tipicamente variabile da 1 a 64K parole (1K = 1024). In breve la memoria principale di un sistema è chiamata normalmente «la memoria». Nel caso di sistemi a microprocessore questa memoria è realizzata con tecnologia MOS/LSI e può persino risiedere direttamente sullo stesso chip di microprocessore. Il dispositivo viene allora detto *microcomputer-on-a-chip*, poichè esso incorpora tutti gli elementi logici di un computer su un unico chip. I microprocessori «standard» richiedono una memoria esterna. Le velocità tipiche sono nella gamma di 300-600 ns.

3. **La memoria di massa.** In considerazione del costo relativamente alto della memoria principale veloce, non è pratico considerare un sistema utilizzante piú di 32K o 64K di memoria principale. Per questa ragione vengono usati speciali dispositivi periferici per immagazzinare grandi quantità di dati su supporti a basso costo. Le due periferiche piú usate per i sistemi a microprocessore sono il nastro magnetico e il floppy disk. Il nastro magnetico è generalmente un nastro in cassette. Un floppy disk può essere sia un regolare floppy disk o un micro-floppy. Questi dispositivi verranno descritti ad Capitolo 7 (Tecniche di Interfacciamento).

ORGANIZZAZIONE DELLA MEMORIA

Una memoria è organizzata logicamente per parole. Una parola è una unità logica di informazione consistente di 4, 8, 12 o 16 bit (un *bit* è un *binary digit*). Un microprocessore a 8-bit richiede 8 bit per i dati e l'ampiezza di parola per un microprocessore a 8-bit è perciò 8 bit. La sua memoria sarà logicamente strutturata per parole di 8 bit.

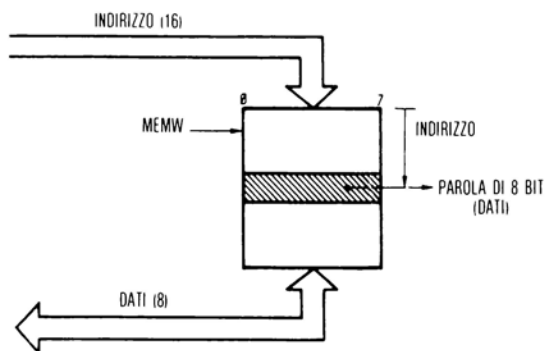



Fig. 1-3: Organizzazione logica di una memoria.

8 bit sono chiamati un *byte*; 4 bit sono chiamati un *nibble* e non esiste nessun'altra parola specifica per altre ampiezze. L'ampiezza di parola di un microprocessore a 8-bit è un byte. Per un processore a 16-bit è 2 byte.

L'organizzazione logica di una tipica memoria appare in Fig. 1-3. L'ampiezza di una memoria è il suo numero di bit cioè la sua ampiezza di parola. I bit sono normalmente ordinati da 0 a n. Nel caso di un microprocessore a 8-bit (che sarà il nastro «microprocessore standard» d'ora in avanti) la posizione del bit entro memoria sarà perciò indicata con una cifra da 0 a 7.

DOMANDA: *Perchè le posizioni dei bit sono indicate con cifre da 0 a 7 invece che da 1 a 8?*

La risposta è che ciascuna cifra da 0 a 7 rappresenta la *posizione binaria* del bit nella parola. Nel sistema binario il right-most digit o bit più alla destra (bit 0) rappresenta 2^0. Il successivo left-most bit o bit piú alla sinistra, bit 1, rappresenta 2^1 e così via.

Verticalmente l'altezza di una memoria è la sua ampiezza o il suo numero di parole. La posizione della parola entro la memoria è chiamata il suo *indirizzo*. La prima parola della memoria ha indirizzo 0, la successiva indirizzo 1 e così via. Per ragioni di decodificazione, la dimensione della memoria è normalmente una potenza di 2. Per esempio: 256, 512, 1K, 2K, 4K parole.

DOMANDA: *Qual'è l'indirizzo dell'ennesima parola della memoria?*

RISPOSTA: *n − 1*

Al fine di leggere i contenuti di una parola entro la memoria è necessario specificare il suo indirizzo. Ciascun modulo di memoria è perciò connesso all'address-bus, o bus degli indirizzi. Un tipico address-bus include 16 linee così che esso può specificare fino a $2^{16} = 64K$ locazioni del dispositivo. Se la reale dimensione di memoria è piú piccola di 64K sarebbe necessario un minor numero di linee. Al fine di selezionare una parola nella memoria verrà inviata sull'address-bus una configurazione di bit che specificherà l'indirizzo della parola desiderata nella memoria. I bit provengono dall'address-bus e sono diretti verso le decodifiche. Una *decodifica* (decoder) seleziona una parola entro la memoria. In risposta a un segnale di controllo, come un READ, per un'operazione di lettura, o WRITE, per un'operazione di scrittura, verrà letta o scritta una parola da o entro la memoria. Nel caso di un'operazione di lettura una parola verrà reperita dalla memoria: dopo un tempo chiamato *tempo di accesso* (access-time), essa sarà resa disponibile sui piedini (pin) di uscita del chip di memoria. Questi pin sono connessi al data bus (8 bit poichè le parole sono ampie 8 bit nel nostro esempio).

DOMANDA: *Perchè solo 8 bit di dati escono da una memoria che sta ricevendo 16 bit di indirizzo?*

RISPOSTA: *Questo è un punto importante da chiarire. Non esiste nessuna relazione diretta tra il numero di bit uscenti dalla memoria come dati e il numero di bit di indirizzo. I bit di indirizzo specificano una posizione entro la memoria e sono usati per selezionare una locazione di parola mediante speciali decodifiche. La parola di dati corrispondente alla posizione selezionata può essere di lunghezza arbitraria, da 1 a p bit. Ad esempio un piccolo chip di memoria potrebbe includere solamente 64 parole x 8 bit. In questo caso l'address-bus richiesto per selezionare una parola in questa memoria, richiederebbe solamente 6 linee ($2^6 = 64$). Tuttavia per ciascuno degli specificati 64 indirizzi che potrebbe essere trasmesso su questo address-bus ridotto, ci sarebbero ancora 8 bit di dati sul data bus.*

Nel caso di un'operazione di WRITE la sequenza è analoga: un indirizzo è specificato sull'address-bus e i dati sono presentati sul data bus. Alla memoria viene specificato un ordine (attraverso il control bus) e la memoria scrive nella locazione di memoria specificata i contenuti presentati sul data bus (nel nostro esempio 8 bit). Il tempo richiesto per scrivere i dati nella memoria è chiamato *tempo di ciclo di memoria* (memory cycle time).

In pratica la maggior parte di programmi su microprocessori richiedono meno di 4K parole. L'indirizzamento di 4K parole richiede 12 bit. Da 4 a 5 linee dell'address bus non sono perciò utilizzate per l'indirizzamento della memoria.

Questa caratteristica verrà utilizzata più avanti per selezionare altri dispositivi connessi al bus del sistema, come quelli di input/output. L'address bus non è utilizzato esclusivamente per indirizzare la memoria, ma è utilizzato per indirizzare qualsiasi dispositivo o più precisamente qualsiasi registro entro qualunque dispositivo che possa essere connesso ai bus del sistema.

Dobbiamo introdurre qui diverse parole di avvertimento. Una memoria a 4 K è una memoria che contiene 4K parole. Un chip di memoria a 4K, d'altra parte, si riferisce a un chip contenente 4K *bit*, non 4K *parole*. Quando ci si riferisce a un componente (1 chip), K si riferisce ai bit; quando ci si riferisce a un sistema di memoria, K si riferisce tradizionalmente alle parole (8 o 16 bit per esempio, in funzione del processore).

Una istruzione contenuta in memoria non occupa necessariamente una sola parola ma può occupare una o più parole (una, due, o tre). Le istruzioni tipiche richiederanno perciò 8, 16, o 24 bit di memoria. Per poter leggere una istruzione a due parole entro l'unità di controllo del microprocessore, saranno richieste due successive operazioni di lettura della memoria.

Il bit a destra (right-most bit) della memoria, generalmente indicato come bit 0, è chiamato bit meno significativo (least-significant-bit), o LSB poichè contiene il più piccolo peso nella rappresentazione binaria dei dati. Il bit più a sinistra, generalmente denominato bit 7, è il bit più significativo (most-significant-bit), MSB. L'LSB e l'MSB giocano un ruolo speciale in vista delle limitazioni di test nella maggior parte dei microprocessori. La maggior parte dei microprocessori può eseguire

solamente test diretti sul valore del left-most bit nel loro accumulatore (l'MSB). A questo bit spetta il ruolo di essere il *bit di segno* (sign bit) nella notazione *complemento a 2* (ciò verrà spiegato più tardi al Capitolo 8 sulla programmazione). Il bit 0, l'LSB, è il secondo bit in ordine di convenienza per noi poichè esso può venir testato entro l'ALU dopo un'operazione di singolo shift a destra. Per questa ragione le informazioni di stato dai dispositivi di ingresso o uscita generalmente compaiono nella posizione di bit 7 di un registro o altrimenti nella posizione di bit 0.

DEFINIZIONI FONDAMENTALI RIGUARDO AL MICROPROCESSORE

Le abbreviazioni e le definizioni fondamentali impiegate per i microprocessori appaiono nell'Appendice alla fine del libro. I termini più importanti saranno definiti.

Hardware, Firmware e Software

L'*hardware* si riferisce ai componenti fisici di un sistema. Il *software* si riferisce ai programmi. Il *firmware* si riferisce ai microprogrammi. Per estensione il termine firmware è anche applicato frequentemente ad ogni programma che risieda in una memoria a sola lettura, cioè che non possa essere cambiata (combinazione di hardware e software). Un *microprogramma* non è un programma per un microprocessore.

DOMANDA: *Qual'è il termine corretto per un programma adatto per un microprocessore?*

RISPOSTA: *Un programma adatto per un microprocessore è chiamato «programma». Non esiste nessuna differenza significativa tra un programma adatto per un microprocessore e un programma per un minicomputer a parte il fatto che un programma per un microprocessore di solito è più difficile da scrivere e può controllare componenti diversi da quelli che controlla un minicomputer. Tuttavia le tecniche di programmazione sono essenzialmente simili, anche se nel caso dei microprocessori è richiesta una maggior conoscenza dell'hardware.*

Un *microprogramma* è un programma di coordinazione per unità di controllo di qualsiasi processore. Tipicamente un microprogramma interpreta il *set di istruzioni* di una «macchina» (una macchina significa qualsiasi computer).

Il *set di istruzioni* è la lista di istruzioni disponibili al programmatore per dare ordini alla macchina, come: «Somma Registro 1 a Registro 2». Si deve notare che un microprogramma è fondamentalmente diverso da un normale programma scritto dall'utilizzatore.

Tipicamente in un microprocessore monolitico (cioè un microprocessore in un unico chip) il microprogramma svolge il coordinamento dell'unità di controllo al fine di definire il set di istruzioni desiderato. (I dispositivi *bit-slice* realizzano solo una fetta di processore da cui è escluso il controllo).

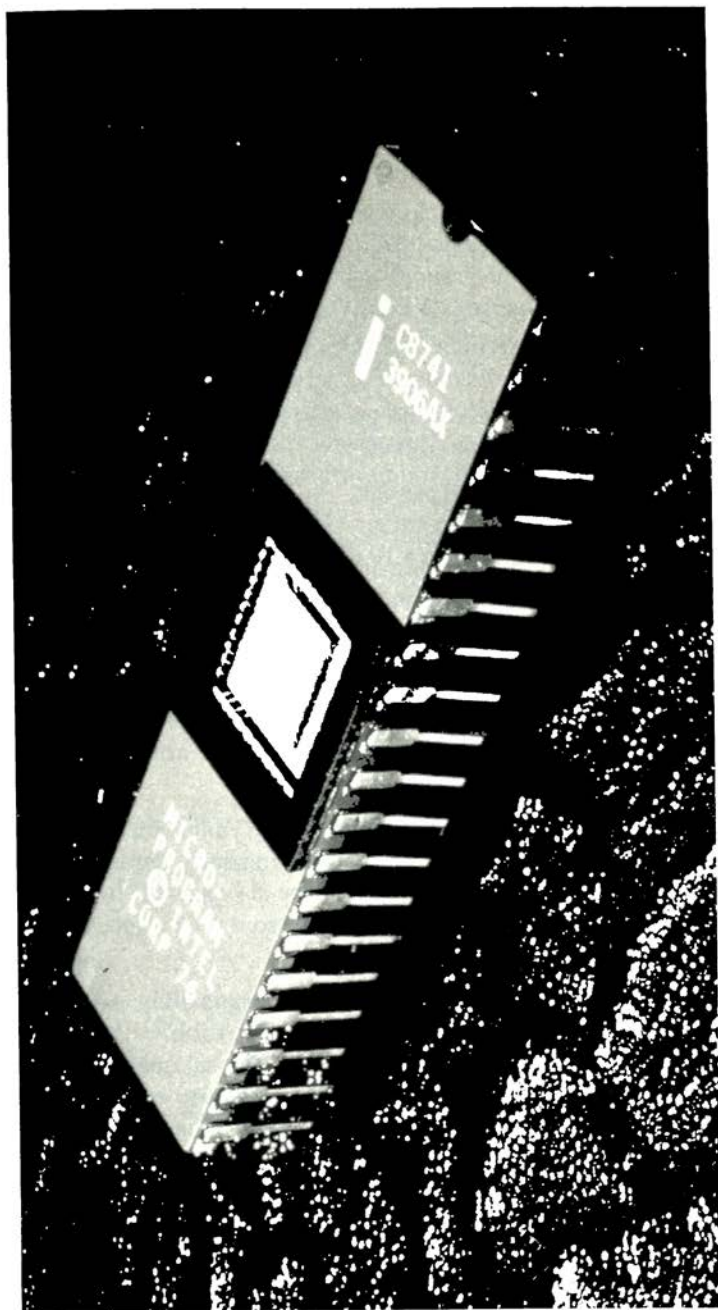



Fig. 1-4: Un microprocessore: chip, DIP, pins.

Large - Scale - Integration (Integrazione a larga scala).

La tecnologia LSI (Integrazione a Larga Scala) apparve alla fine degli anni 60 ed oggi permette di realizzare da 1 a 15.000 transistori su un unico chip. L'attuale processo di fabbricazione sarà descritto più avanti in questo capitolo.

Fin dalla nascita del transistore attorno al 1946-47, la densità di componenti per millimetro quadrato sono aumentate costantemente. Il primo circuito (IC) apparve, quindi la tecnologia SSI (Small-Scale-Integration che significa Integrazione su piccola scala) quindi la tecnologia MSI (Medium-Scale-Integration) e in fine la tecnologia LSI (Large-Scale-Integration), usata proprio al giorno d'oggi. La tecnologia è ora in rapido sviluppo verso la VLSI (Very-Large-Scale-Integration che significa Integrazione su Grandissima Scala) e presto la SLSI (Super-Large-Scale-Integration).

Sebbene non esistano limiti definiti tra SSI, MSI, LSI e VLSI quella che segue è una classificazione molto approssimata:

SSI = da 1 a 10 transistori per chip
MSI = da 10 a 100-500 transistori
LSI = da 100-500 transistori a 10.000-20.000
VLSI = oltre 10.000 transistori
SLSI = oltre 50.000 transistori

Il Microprocessore

Il microprocessore LSI può essere ora ridefinito: un microprocessore è un componente LSI che realizza le funzioni di un'unità logica-aritmetica più la sua unità di controllo associata in un unico chip.

In pratica la maggior parte dei microprocessori «monolitici» richiedono almeno 2 e talora 3 componenti per realizzare queste funzioni. Per esempio l'Intel 8080 non richiede solamente la MPU 8080 ma, al fine di svolgere le funzioni di una CPU richiede il circuito di clock 8224 più il suo quarzo e il «controller del sistema» (system - controller) 8228.

La fotografia di un microprocessore reale è riportata il Fig. 1-4. Il coperchietto è stato rimosso per mostrare il *chip* che stava al di sotto.
Il *chip* è un pezzetto rettangolare di silicio (o *die*) sul quale è stato realizzato il circuito. Il chip è saldato (bonded) al *package* (in questo caso un DIP = Dual In-Line Package, che significa Contenitore con i piedini allineati in doppia fila). I pins (piedini) del package possono essere inseriti direttamente nei fori su una piastra di circuito stampato o su un altro mezzo. I microprocessori tipici hanno da 40 a 42 pins al massimo. Ciò non è dovuto ad una limitazione fisica sui packages, ma ad una limitazione economica. (I testers industriali sono limitati a DIP fino a 40 pin; il costo di acquisto di un tester è di 500.000$ o più).

I rettangoli bianchi che appaiono su entrambi i lati del chip entro la porzione

scoperchiata del package sono i *pad*. Questi pad sono collegati mediante un circuito stampato interno, entro il package, ai pin. Un sottile filo d'oro è saldato ai pad appartenenti al package e li connette ai pad sulla periferia del chip stesso. Il chip è perciò elettricamente collegato attraverso i fili d'oro, attraverso i pad e il circuito stampato ai pin del DIP.

COSTRUZIONE DI UN MICROPROCESSORE

In questo paragrafo verrà spiegata la *tecnologia MOS* (MOS = Metal Oxide Semiconductor). In breve i transistori ed altri componenti sono realizzati sulla superficie di pezzetti di silicio chiamati chip.

Per realizzare un circuito MOS si deve prima accrescere un monoscristallo di silicio. Questo cristallo viene successivamente affettato in fette sottilissime circolari chiamate *wafer*. Il cristallo viene tagliato in una direzione specifica del reticolo cristallino. Da un unico wafer verranno create diverse dozzine di chip. Un «chip» non ancora distaccato dal wafer è chiamato *die* durante il processo di fabbricazione. Questi chip diventeranno chip di microprocessori, chip di memoria, o di altri moduli. I dice (plurale di die) vengono creati nel wafer mediante un processo fotolitografico, analogo alla stampa usuale delle fotografie. Mediante iniezione di impurità (*doping*) e utilizzando un processo di mascheratura e diffusione vengono create aree positive e negative entro il silicio. Questo processo verrà spiegato nei dettagli al

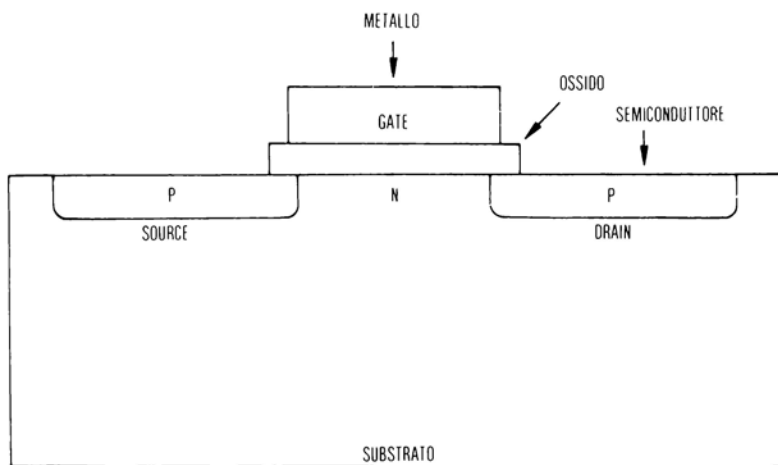



Fig. 1-5: Transistore a canale p.

prossimo paragrafo. Allorquando su un wafer sono stati creati dei dice completi, il wafer è testato su speciali punti di test che sono stati formati specificamente sui bordi e al centro del wafer. Se questi test hanno successo il wafer verrà inciso e spezzettato in dice individuali. Questi dice sono allora chiamati chip. Ciascun chip

verrà poi montato su un package, connesso mediante fili d'oro ai pad del package e verrà sottoposto ad ulteriori test visivi, elettrici ed ambientali. Il package verrà poi sigillato e sarà sottoposto a test finali. I package sigillati che hanno superato con successo i test di produzione possono essere allora commercializzati.

Fabbricazione di un transistore PMOS

PMOS si riferisce alla tecnologia a canale P. Gli altri tipi di tecnologie MOS attualmente già sviluppate verranno descritte nei prossimi paragrafi.

In Fig. 1-5 è riportato un tipico transistore PMOS. Esso utilizza silicio di tipo n che verrà drogato con impurità di tipo p per creare il source e il drain del transistore. Gli elementi di drogaggio sono boro e fosforo. Le aree da drogare con impurità vengono delimitate con una maschera. Il processo corrente utilizzato per definire queste aree è quello fotolitografico. Un nuovo processo tuttora in fase sperimentale è quello a raggi di elettroni, attualmente in sviluppo. Nella maggior parte dei casi le impurità sono iniettate nell'area esposta del silicio mediante diffusione termica. Viene anche usato un processo di implantazione ionica per ottenere un più preciso allineamento del gate riducendo in tal modo la capacità parassita (ciò risulta in una aumentata velocità di commutazione). Tuttavia è più costosa e non può essere rea-

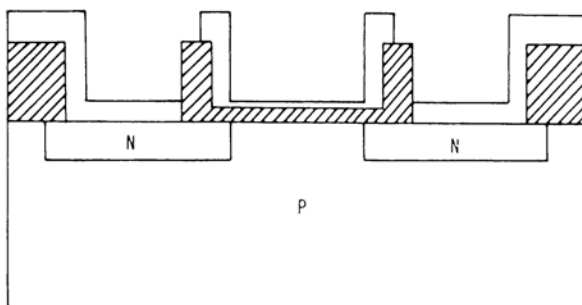



Fig. 1-6: Transistore a canale n.

lizzata su grande serie richiedendo in più un acceleratore. Al centro di Fig. 1-5 compare il gate del transistore. Esso è stato realizzato depositando uno strato di metallo (sia silicio o alluminio - con ciò ne risulta un transistore silicon gate o aluminium gate). Il gate è isolato dal substrato da un deposito di biossido di silicio che è stato accresciuto su di esso.

Al gate verrà applicata la polarizzazione negativa rispetto al source. La presenza di cariche negative nel gate determinerà l'apparizione di cariche positive (buchi o lacune) nel *canale* tra il source e il drain. Ci sarà allora un canale di conduzione tra il source e il drain. Il transistore sarà polarizzato «on». Questo è un transistore a canale p o transistore PMOS. Similmente la struttura di un transistore a canale n compare in Fig. 1-6. La reale fabbricazione verrà ora descritta con maggior detta-

Fig. 1-7: Il substrato di silicio.

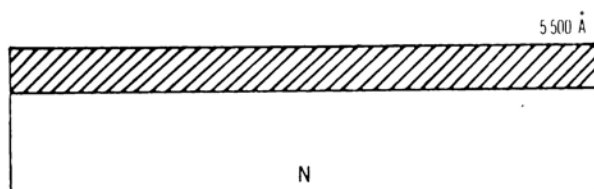



Fig. 1-8: La prima ossidazione genera uno spesso strato di SiO_2.

glio. Il wafer di silicio verrà ottenuto tagliando un monocristallo nella direzione appropriata, per esempio direzione 111 del reticolo (Fig. 1-7). Successivamente viene depositato sul silicio uno strato spesso di ossido (da 5.000 a 6.000 Å. Vedi Fig. 1-8. 1Å = 10^{-9} cm).

Come prima operazione verrà utilizzato una prima maschera per definire le zone p entro il silicio. Esse saranno le aree di source e drain del transistore. Sull'ossido di silicio viene depositata un'emulsione fotosensibile e mediante una maschera viene impressionata l'area che sarà drogata. L'ossido che ricopre le aree che dovranno essere drogate sarà quindi asportato mediante etching (corrosione) chimico (Fig. 1-9). Sulle aree esposte può allora essere eseguito il drogaggio e le impurità di tipo p sono introdotte nel silicio in genere mediante un processo di diffusione termica (Fig. 1-10). Sulla superficie del silicio viene accresciuto ancora una volta uno strato spesso di ossido (da circa 10.000 a 15.000 Å — vedere Fig. 1-11). Una nuova maschera viene impiegata per definire le aree che saranno più tardi metallizzate. Quindi l'ossido viene rimosso da queste ultime posizioni (Fig. 1-12). Un'ultima ossidazione, detta ossidazione del gate, viene successivamente eseguita per accrescere un sottile strato d'ossido (da 1000 a 1500 Å — vedere Fig. 1-13). Una terza fase fina-

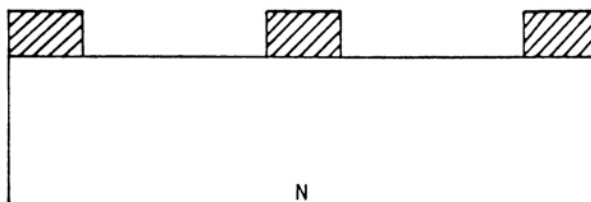



Fig. 1-9: L'attacco rimuove l'ossido.

le di asportazione dell'ossido viene eseguita per esporre le aree di source e drain che dovranno essere collegate al resto del circuito dopo la fase di metallizzazione (vedere Fig. 1-14).

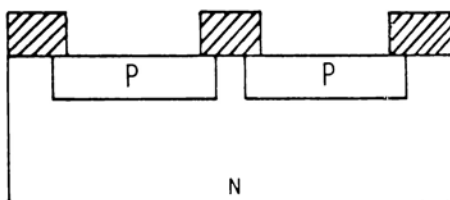



Fig. 1-10: Il drogaggio viene eseguito per diffusione termica.

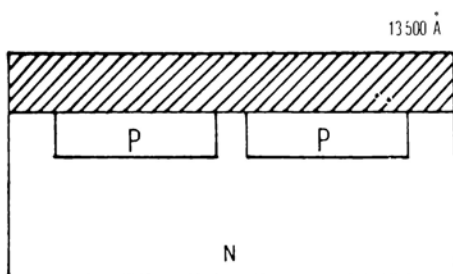



Fig. 1-11: Seconda ossidazione: ossido spesso.

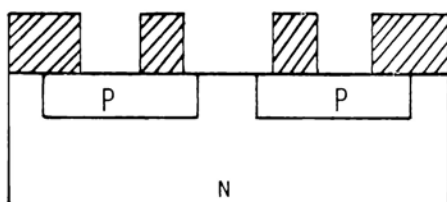



Fig. 1-12: Seconda rimozione di ossido.

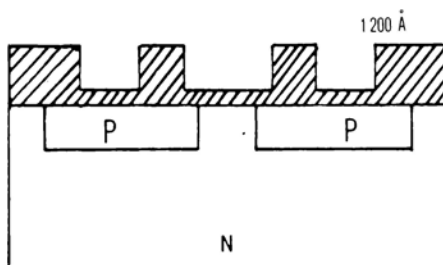



Fig. 1-13: Ossidazione di gate.

La fase finale è la fase di metallizzazione durante la quale viene generalmente depositato dall'alluminio sulle aree esposte, collegando il source, il gate e il drain agli altri componenti del circuito. I transistore è ora completato.

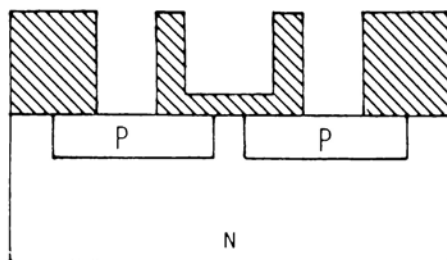



Fig. 1-14: La terza rimozione di ossido apre le aree di source e drain.

TECNOLOGIE LSI

Si possono distinguere sei categorie principali di tecnologie LSI:

1. Tecnologia PMOS

La fabbricazione di un transistore a canale P è stata appena descritta. I transistori a canale P utilizzano la mobilità delle caratteristiche positive, chiamate buchi, o vacanze, per condurre elettricità. Quella PMOS è una tecnologia MOS relativamente vecchia, attualmente ben compresa e che è diventata economica. È stata ampiamente usata per tutti i primi microprocessori. Essa fornisce un'eccellente densità (da 3.000 a 15.000 transistori per chip). Tuttavia essa risulta lenta se paragonata alle tecnologie più nuove, quali la NMOS. Una delle principali ragioni di preferenza per i costruttori deriva dal fatto che il processo è ben conosciuto e che ci si può spingere verso forti aumenti di complessità con un'alta probabilità di successo.

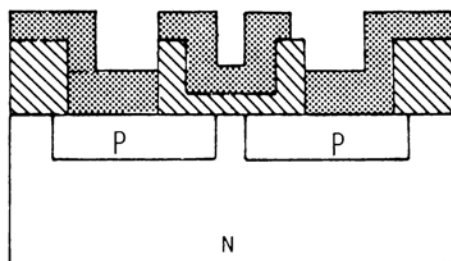



Fig. 1-15: Metallizzazione finale (Al).

2. Tecnologia NMOS

La NMOS è intrisecamente più veloce della PMOS perchè utilizza elettroni piuttosto che buchi come portatori di carica. Fornisce un'eccellente densità ed è probabilmente il miglior compromesso per realizzare oggi un microprocessore veloce e complesso. Tuttavia, essendo una tecnologia recente, non è ancora cosi ben sviluppata come la PMOS è non è usata da tutti i costruttori. Un tipico microprocessore NMOS raggiunge velocità di esecuzione di istruzioni dell'ordine di 1 microsecondo. Esso risulta tipicamente due volte più veloce di un equivalente microprocessore PMOS.

3. Tecnologia CMOS (MOS Complementare)

La tecnologia CMOS utilizza una combinazione di transistori l'uno a canale p e l'altro a canale n. Le caratteristiche della tecnologia CMOS sono perciò a metà strada tra quelle delle tecnologie NMOS e PMOS. La CMOS è più veloce della PMOS, ma un poco più lenta della NMOS e raggiunge una buona densità. Tuttavia, usando due transistori invece di uno, permette una più bassa integrazione dell'NMOS. Il vantaggio fondamentale è il suo bassissimo consumo di potenza e l'eccellente immunità al rumore (quasi ideale: 40% di immunità al rumore – i dispositivi CMOS possono funzionare in una fascia da 2V a 12V). La tecnologia CMOS fu specificamente creata per applicazioni areonautiche e spaziali.

Ora è utilizzata essenzialmente in sistemi che richiedono la portatilità e/o un bassissimo consumo di potenza. I due microprocessori commerciali che utilizzano questa tecnologia sono il Cosmac dell'RCA e l'Intersil 6100. La struttura di un circuito CMOS appare in Fig. 1-16.

4. Tecnologie bipolari

La tecnologia bipolare è una delle tecnologie più veloci disponibili oggi. Tra le tecnologie bipolari quella più usata è probabilmente la low-power-Shottky TTL (LPSTTL). Questa tecnologia è utilizzata oggi per la realizzazione di dispositivi ve-

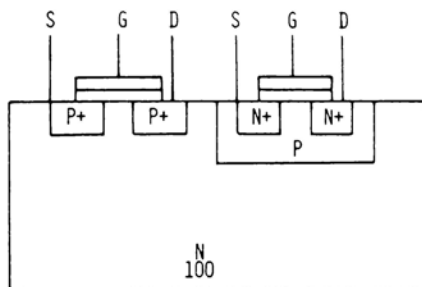



Fig. 1-16: La tecnologia CMOS comprende 2 tipi di transistori.

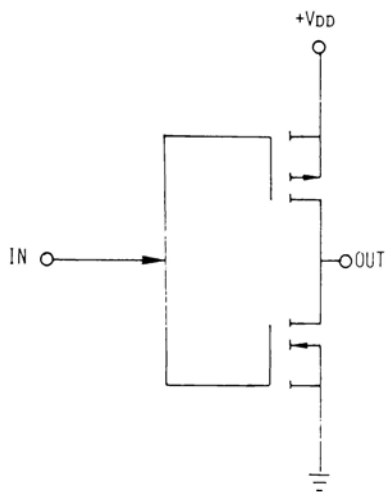



Fig. 1-17: Circuito CMOS di base.

loci bit-slice che raggiungono velocità da 70 a 100 ns per istruzioni rispetto a circa 1 μs (microsecondo) per un microprocessore monolitico. I due principali svantaggi della tecnologia bipolare sono il sostanziale consumo di potenza, la dissipazione di potenza conseguente e la sua bassa densità. Non è perciò ancora possibile realizzare un completo microprocessore monolitico in un singolo chip. In un singolo chip possono essere realizzate solamente delle frazioni o fette (slices) di una CPU tradizionale e questi sono i nuovi dispositivi bit-slice che verranno descritti alla fine del Capitolo 3. La caratteristica essenziale dei bipolari è la loro alta velocità. La realizzazione di un transistore bipolare appare in Fig. 1-18. Entro il campo delle tecnologie bipolari l'ECL mantiene una posizione potenziale per altissime velocità.

La tecnologia I2L è una tecnologia bipolare ma merita una trattazione particolare. I2L significa «Integrated Injection Logic» o logica integrata ad iniezione. La tecnologia I2L fu sviluppata come risultato delle applicazioni nei mercati delle calcolatrici tascabili e degli orologi. Essa è caratterizzata da velocità da bipolare (solo in linea di principio — fino ad oggi) ed un basso consumo di potenza, critico per le applicazioni portatili menzionate. L'I2L ha raggiunto un bassissimo consumo di potenza ed è ora utilizzata nel mercato consumer.

Tuttavia la velocità caratteristica dei dispositivi bipolari non è ancora stata raggiunta. L'I2L è stata usata in particolare per la realizzazione di dispositivi bit slice dalla Texas Instruments. Essi non possono ancora competere in velocità con i loro equivalenti TTL. Un altro vantaggio dichiarato per l'I2L è l'alta integrazione che può essere ottenuta. Allorchè l'alta integrazione e l'alta velocità diverranno una realtà, l'I2L potrà diventare una tecnologia importante per i microprocessori per applicazioni portatili.

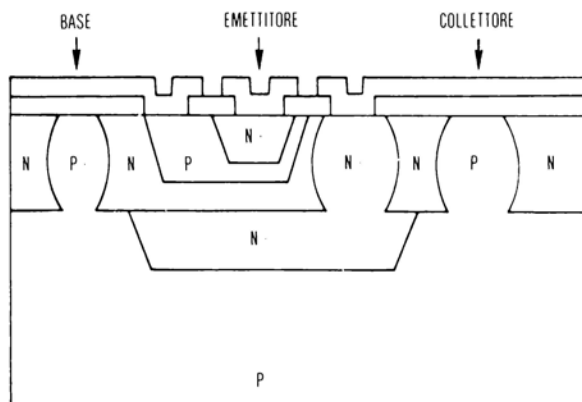



Fig. 1-18: Un transistore bipolare.

5. CCD

I dispositivi ad accoppiamento di carica o charge-coupled devices (CCD) sono usati per rendere disponibile una nuova forma di memoria ad altissima densità. È già possibile realizzare un chip da 64K-bit usando la tecnologia CCD. Il principio è il seguente: sull'ossido di silicio vengono depositati dei quadratini di alluminio a distanza regolare. Ciascuno di questi quadratini immagazzinerà una carica. Per merito della loro semplicissima geometria un gran numero di questi quadratini può essere depositato su un singolo chip. Tuttavia, come tutti i condensatori, questi condensatori elementari hanno delle perdite ed ogni carica deve essere rinfrescata. Le cariche sono rinfrescate mediante uno shift circolare di cariche da un quadratino di alluminio al successivo. Questa memoria è perciò una memoria a circolazione. Essa è stata chiamata memoria di tipo a tamburo per l'analogia con un tamburo magnetico rotante. Si considera che se i prezzi delle memorie CCD continueranno a scendere esse diverranno competitive in pochi anni con i floppy disks.

6. Altre tecnologie

A tuttoggi esistono diverse altre tecnologie che sono state usate da alcuni costruttori in casi speciali:

Memorie a bolle (Bubble Memories) hanno cominciato solo ora ad apparire ed offrono altissime densità. Tuttavia esse sono tuttora costose e lente. Nel futuro esse potrebbero presentarsi come serie concorrenti alle memorie di tipo disco (la bubble memory è non volatile).

MNOS è stata usata per realizzare memorie EAROM (Electrically Alterable Read-Only Memories che significa ROM alterabili elettricamente). Questi tipi di ROM

verranno descritte al Capitolo 3. MNOS (Metal-Nitride-Oxide-Semiconductor) non deve essere confuso con NMOS (MOS a canale N).

DMOS (Double-Diffused MOS che significa MOS a doppia diffusione) è stata usata per memorie dinamiche ad alta densità.

VMOS (viene utilizzato un solco con sezione a V) viene pure usata per le memorie dinamiche RAM ad alta densità onde ottenere una ancora più alta densità (n.d.t. e tempi di accesso ridotti).

UNA BREVE STORIA DEI MICROPROCESSORI

L'avvento dei microprocessori non è dovuto a previsione, astuzia di progetto o pianificazione avanzata. Esso è stato accidentale. I primi microprocessori introdotti sul mercato sono stati errori della tecnologia e spesso scarti. Questo fatto è fondamentale per la comprensione dei prodotti a microprocessore contemporanei. A causa dell'introduzione disordinata e non pianificata dei microprocessori i loro errori di progetto e inadeguatezze sono state propagandate fino ad oggi nel nome delle compatibilità. Molte delle «peculiarità» caratteristiche dei microprocessori di oggi sono il risultato di questo background. Perciò è importante a questo punto ritracciare brevemente la storia dei microprocessori.

Il primo reale avvento di transistori funzionanti può essere rintracciato indietro verso la fine degli anni 40, dopo la guerra. Fu solo 10 anni più tardi che il primo modello funzionante di un circuito integrato fu sviluppato da St. Clair Kilby della Texas Instruments. Circa nello stesso periodo fu sviluppato il processo planare da Hoerni alla Fairchild (1959). Attorno al 1961 i primi IC iniziarono ad essere prodotti in quantità.

Di lì l'integrazione di circuiteria proseguì rapidamente. La SSI (Small-Scale-Integration) apparve nel 1964. Essa è un gate completo in un singolo chip. (Un gate è, ad esempio, un «AND logico» «OR» «NOT»); esso richiede diversi transistori. Nel 1968 compare la MSI (Medium-Scale-Integration): essa consiste in un registro completo su un singolo chip.

L'LSI commerciale (Large-Scale-Integration) compare nel 1971: la prima memoria da 1K-bit, la prima UART e il primo microprocessore. Il primo microprocessore di uso generale fu introdotto nel tardo 1971 (Intel 4004).

La ricerca verso una più alta integrazione derivò inizialmente da contratti con il governo, in particolare dalla NASA (contratti aereospaziali) ed altri programmi militari. In questi programmi il costo non fu un fattore essenziale. Il criterio essenziale fu la miniaturizzazione. È all'inizio del 1970 che i contratti di ricerca dalla NASA e dal governo diminuiscono rapidamente spingendo i costruttori a cercare sorgenti alternative di finanziamento. È anche in questo periodo che i costruttori di circuiti integrati scoprono per la prima volta uno sbocco dei loro prodotti verso il pubblico: questo sbocco si concretizza nelle prime calcolatrici da tavolo che saranno seguite dalle calcolatrici tascabili. Per la prima volta è possibile vendere LSI al

pubblico. La produzione e vendita di centinaia di migliaia di unità ora divenuta una realtà.

Nella prima parte del 1971 furono sviluppati solo due prodotti LSI standardizzati. La prima RAM dinamica da 1K-bit e la prima UART (Universal-Asynchronous-Receiver-Transmitter-un convertitore serie/parallelo). A quei tempi nessuno poteva immaginare quali potessero essere i successivi prodotti standard. Fu allora che i microprocessori furono introdotti sul mercato con i risultati che ora noi sappiamo.

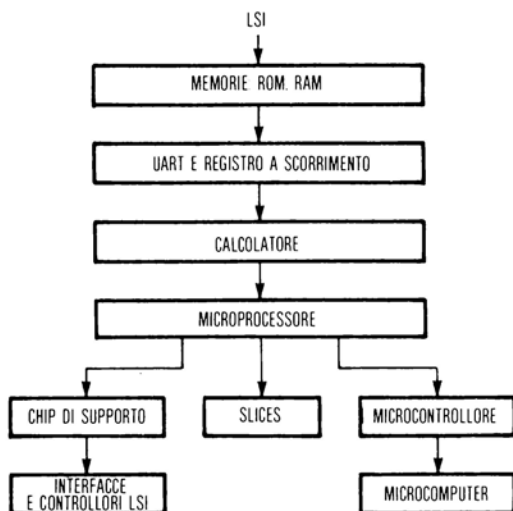



Fig. 1-19: L'evoluzione della tecnologia LSI.

Il progetto dell'Intel 4004 iniziato nel 1971 derivò da un contratto con un costruttore giapponese di calcolatrici da tavolo. In realtà i primi acquirenti del 4004 (un microprocessore a 4-bit) dovettero firmare un contratto che li impegnava a non sviluppare un calcolatore da tavolo con il chip per almeno un anno. Questo primo microprocessore di uso generale fu di fatto progettato come un calcolatore.

Esso era perciò non potente e generalmente inadeguato per calcoli di uso generale. Non c'erano grandi aspettative circa la sua possibilità di essere venduto.

Il successivo evento significativo fu l'introduzione dell'8008, dalla Intel, nel 1972 che fu il primo microprocessore a 8-bit di impiego generale. Alcuni anni fa la Display Terminals Corporation, ora conosciuta come Datapoint generò una richiesta di gara per la produzione di un processore monolitico capace di controllare un CRT in un singolo chip. Due società concorsero per l'ottenimento e lo sviluppo del

contratto: la Texas Instruments e la Intel. Dopo mesi di tentativi la Texas Instruments si ritirò. L'Intel continuò lo sviluppo ed uscì con un componente che poteva fondamentalmente soddisfare tutte le richieste funzionali della Datapoint eccetto una: era troppo lento. In più circa nello stesso tempo era esplosa una guerra sui prezzi dei componenti bipolari che portò a una riduzione significativa dei prezzi per i prodotti bipolari.

In considerazione della velocità inadeguata del processore sviluppato dalla Intel, e dei prezzi molto più bassi dei componenti bipolari, la Datapoint decise di realizzare il proprio controllore con tecnologia bipolare. L'Intel fu lasciata con un chip il cui sviluppo era stato pagato ma per il quale non esisteva nessun mercato ovvio. Un prodotto fondamentale della Intel, allora una società molto giovane, era costituita dalle memorie. L'8008 fu introdotto perciò (con riluttanza?) sul mercato con la speranza che esso permettesse di vendere un maggior numero di chip di memoria. Sarebbe sembrato che tutti gli sforzi di progetto fossero stati dunque bloccati e il gruppo di progettazione assegnato ad altri compiti. Quella avrebbe dovuto essere la fine dei microprocessori!

Con presumibile sorpresa del costruttore (e dei suoi concorrenti) le vendite di questo nuovo prodotto (il microprocessore) iniziarono ad aumentare rapidamente. Uno dei costruttori era incappato accidentalmente sul successivo prodotto LSI del mercato: il microprocessore. L'Intel comprese rapidamente il potenziale di questo nuovo dispositivo, ricostruì un gruppo di progettazione e introdusse un anno più tardi il successore dell'8008, l'8080. Simultaneamente tutti i concorrenti principali si impegnarono a sviluppare le proprie versioni di ciò che l'8080 avrebbe dovuto essere se fosse stato progettato correttamente fin dall'inizio. Entro i due anni che seguirono tutti i principali microprocessori «standard» ora sul mercato sarebbero stati introdotti, la maggior parte di loro ispirati dal progetto iniziale dell'8080. La Motorola introdusse il 6800 (circa un anno dopo l'8080), la Rockwell introdusse il PPS8, la Signetics il 2650, ecc.

Siamo ora entrati in una terza generazione nel progetto del microprocessore nella quale i successori all'8080 ed al 6800 sono stati posti sul mercato. Essi sono lo Z 80 della Zilog, l'8085 della Intel e, i nuovi microcomputers in un chip «L'F8 della Fairchild e della Mostek, l'8048 della Intel, il PPS4 della Rockwell, il TMS 1000 e il 9940 della Texas Instruments). Una valutazione oggi sul mercato è presentata al Capitolo 3.

LA SILICON VALLEY

Da un punto di vista storico si può considerare generalmente che l'industria dell'LSI, così come può essere vista oggi, nacque da due poli di sviluppo negli Stati Uniti, rispettivamente l'area di Boston sulla Costa Orientale (Bell Telephone Labs) e la regione attorno a Sunnyvale (a sud di San Francisco), ora conosciuta come «Silicon Valley». Molte se non la maggior parte, delle società che popolano la Silicon

Valley sono familiarmente considerate come se fossero nate dalla Fairchild. Come esempio si può ritracciare la storia di alcuni dei progettisti di dispositivi meglio conosciuti oggi.

Shockley (il fisico Premio Nobel) ha messo insieme un piccolo gruppo di scienziati sulla Costa Orientale circa 15 anni fa. A questo gruppo appartenevano Robert Noyce e Gordon Moore. Shockley lasciò la Bell Telephone al fine di creare la propria società, la Shockley Research Laboratories. Alcuni anni più tardi un piccolo gruppo di scienziati avrebbero abbandonato questa società e ne avrebbero creata una nuova la Fairchild, in quella che sarebbe diventata la Silicon Valley. Alcuni anni più tardi un piccolo gruppo di scienziati avrebbero ancora abbandonato la Fairchild e avrebbero creato un'ennesima nuova società a Sunnyvale, «Integrated Electronics» (Intel). La Intel fu fondata nel 1968 da Robert Noyce e da Gordon Moore e avrebbe introdotto tre anni più tardi il primo «microprocessore», con il successo conosciuto oggi. È interessante notare che 5 anni fa tre dei principali progettisti dell'8080, uno dei più popolari microprocessori a 8-bit nel 1976, abbandonarono la Intel per creare la loro propria società: la Zilog. La Zilog ha introdotto oggi un successore all'8080, lo Z80, che compete direttamente con i prodotti della Intel. La storia non ci dice ancora se qualcuno dei progettisti di punta dei prodotti Zilog intenda separarsi da questa società al fine di creare la sua propria nella Silicon Valley. Questo meccanismo di creazione e di implantazione di società è stato tipico nel campo dell'elettronica e in particolare in quello dei circuiti integrati. Ciò contribuisce al fatto che molti dei principali microprocessori oggi sul mercato abbiano caratteristiche così tremendamente simili (ed inadeguatezze di progetto).

VANTAGGI DEI MICROPROCESSORI

I microprocessori stanno spostando l'elettronica tradizionale da una posizione sostanziale delle sue precedenti aree di applicazione e contemporaneamente trovano impiego in quasi tutte le aree che coinvolgono programmi o controlli automatici. I due speciali ed esclusivi vantaggi dei microprocessori sono:

1. Minor numero di componenti

Il piccolissimo numero di componenti richiesti da un sistema a microprocessore si traduce in diversi vantaggi:

- volume ridotto e miniaturizzazione (portatilità) del sistema;
- ridotto consumo di potenza;
- ridotta dissipazione di potenza;
- il minor numero di interconnessioni si traduce in un'aumentata affidabilità;
- infine le caratteristiche menzionate danno come risultato dei costi sostanzialmente più bassi. Un tipico microprocessore di impiego generale viene venduto, oggi, per circa $ 10 in quantitativi di 100 pezzi e per $ 1 o $ 2 in quantitativi di 10.000 pezzi (da $ 15 a $ 30 × per pezzi singoli).

	LOGICA CABLATA	MINICOMPUTER	CHIP-CUSTOM	MICROPROCESSORE
PROGETTO	COMPLESSO	PIÙ SEMPLICE	PIÙ COMPLESSO	SEMPLICE
POSSIBILITÀ DI EVOLUZIONE	COMPLESSA	ECCELLENTE	NESSUNA	ECCELLENTE
TEMPO DI SVILUPPO	SCARSO	MOLTO VELOCE	PEGGIORE	VELOCE
COSTO (BASSO VOLUME ELEVATA COMPLESSITÀ)	ALTO	BASSO	MOLTO ALTO	DA BASSO A MEDIO
COSTO (ELEVATO VOLUME E COMPLESSITÀ)	ALTO	ALTO	BASSO	BASSO
PRESTAZIONI	POTENZIALMENTE MIGLIORI	DA MEDIE A BUONE	DA BASSE A MEDIE	DA BASSE A MEDIE
RIDUZIONE DIMENSIONI	SCARSA	PEGGIORE	MIGLIORE	ECCELLENTE
AFFIDABILITÀ	DA DISCRETA A BUONA	BUONA	CRITICABILE	BUONA
COMPLESSITÀ POTENZIALE	SCARSA	ECCELLENTE	DISCRETA	ECCELLENTE
FACILITÀ DI APPLICAZIONE	PEGGIORE	MIGLIORE	BUONA	BUONA

Fig. 1-20: Le quattro scelte — un confronto.

2. Programmabilità

I concetti e le tecniche di programmabilità saranno descritte al Capitolo 8. I vantaggi fondamentali della programmazione consistono nella semplificazione del progetto e nella riduzione del tempo di sviluppo. In breve la programmazione sostituisce una tastiera al saldatore e fornisce potenti mezzi consistenti in programmi di sviluppo al posto di complessi ed inefficienti mezzi di debugging hardware (il debugging è il procedimento di identificazione e di rimozione degli errori).

Inoltre la programmazione permette di ottenere moduli hardware standardizzati. Un sistema a microprocessore standard può essere programmato per svolgere compiti diversi. La sostituzione di un programma al posto di un altro può non coinvolgere alcun cambiamento hardware oppure può semplicemente richiedere la sostituzione di un nuovo chips di memoria contenente il nuovo programma. Il prodotto può allora essere sviluppato rapidamente, provato sul campo e poi raffinato progressivamente, senza la necessità di alcuna riprogettazione hardware (a parte gli errori di progetto). Inoltre si possono inventare e sviluppare nuove funzioni in tempi successivi e generalmente ciò non richiederà alcuna modifica sostanziale dell'hardware. La porzione hardware del sistema potrà allora essere prodotta in serie e standardizzata con il risultato di bassissimi costi. Il costo di generare il software (i programmi) non è piccolo ma generalmente esso è distribuito su un ampio numero di unità. Le applicazioni tipiche dei microprocessori coinvolgono, per questa ragione, almeno 100 o più unità, alfine di suddividere il costo dello sviluppo del software su un largo numero di sistemi. Solamente ove sia essenziale la microminiaturizzazione (applicazioni medicali e militari) il costo di programmazione è essenzialmente libero come nel caso del nuovo mercato degli *hobbisti*. Da un punto di vista concettuale la programmazione è analoga ad una funzione di «intelligenza» inclusa nel sistema. Molte applicazioni a microprocessore sono state ultimamente qualificate co-

me dispositivi *intelligenti.* Mediante l'uso di un microprocessore precedenti dispositivi a logica cablata sono stati rimpiazzati con sistemi a microprocessore che disponevano di un'aumentata intelligenza.

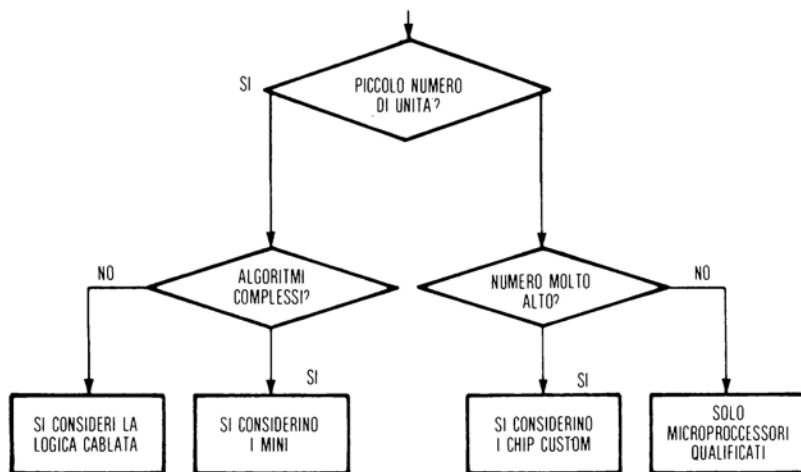



Fig. 1-21: Scelta di un microprocessore.

SOMMARIO

Un microprocessore è una realizzazione LSI ad 1-chip di un tradizionale Central Processing Unit. Fin dal 1977 esso può incorporare non solo la CPU ma anche la memoria e qualche funzione di input/output di un tradizionale computer: esso è un microcomputer-on-a-chip. Esso viene oggi venduto tipicamente per meno di $ 10. Il prezzo sta tuttora scendendo rapidamente e l'integrazione è in aumento. La velocità è approssimativamente per un'istruzione tipica.

Il microprocessore incorpora almeno due delle unità standard di qualsiasi sistema calcolatore: l'*Unità Logica Aritmetica* e l'*Unità di Controllo* che sincronizza il funzionamento del sistema. Il microprocessore dà origine a tre bus:

Il *data-bus,* tipicamente un bus bidirezionale a 8-bit, lungo il quale transiteranno i dati scambiati dagli elementi del sistema.

L'*address-bus,* tipicamente un bus monodirezionale a 16-bit, che si propaga dal microprocessore verso i dispositivi che esso può indirizzare. Il bus degli indirizzi seleziona un registro o un chip entro uno dei dispositivi collegati al sistema.

Il *control-bus,* un insieme di linee da o verso il microprocessore, che realizzano la sincronizzazione ed il controllo dell'intero sistema.

Un microprocessore monolitico necessita di dispositivi di *memoria* e di *input/output*. La memoria viene tipicamente realizzata mediante chip specializzati di memoria, che verranno studiati al Capitolo 3. L'input/output viene realizzato mediante periferiche specializzate di input/output, collegate al sistema da speciali dispositivi di interfaccia che verranno studiati ai Capitoli 3 e 7. Tutti i componenti del sistema sono collegati ai bus. La memoria del sistema contiene i *programmi* che sono un insieme di istruzioni che rappresentano la sequenza di ordini dati al sistema. La memoria contiene anche i *dati* sui quali dovrà operare la ALU.

La soluzione di un problema di controllo, come la sequenzializzazione di una macchina lavatrice, verrà specificato nella forma di una soluzione passo-passo chiamata *algoritmo*. L'algoritmo è la soluzione passo-passo del problema di controllo specificato. Un algoritmo può essere espresso in qualsiasi forma. Poichè nessun calcolatore può comprendere direttamente un linguaggio naturale, tutti gli algoritmi devono essere tradotti in un sottoinsieme ridotto della lingua Inglese che un calcolatore possa capire ed eseguire. Questo linguaggio viene chiamato il *linguaggio di programmazione*. La sequenza delle istruzioni corrispondenti all'algoritmo ed espresse in questo linguaggio è chiamata *programma*. L'insieme di istruzioni che il calcolatore può comprendere direttamente (generalmente espresso in binario) è il *linguaggio macchina*. L'azione di tradurre un algoritmo in un linguaggio di programmazione è la programmazione. Ciò verrà studiato al Capitolo 8.

L'architettura generale di un sistema calcolatore è stata ora stabilita. Si è mostrato come essa sia in relazione con l'architettura di un sistema microprocessore. Si è mostrato come il microprocessore sia un componente LSI che incorpora parte o quasi tutte le funzioni di un sistema calcolatore tradizionale in uno o più componenti. Altri componenti specializzati sono stati introdotti onde semplificare il progetto di un sistema basato su un microprocessore. Essi verranno esaminati a loro volta nei capitoli che seguono. Stabiliremo ora come un microprocessore operi nel suo interno e poi ci apriremo la strada verso l'esterno del microprocessore. Studieremo l'architettura interna di un tipico chip microprocessore, vedremo come esso manipoli le informazioni lungo i suoi bus interni ed usi i suoi *registri*. Vedremo come esso *prelevi* (fetch), *decodifichi* (decode) ed *esegua* (execute) le istruzioni sequenzialmente oppure fuori sequenza. Il meccanismo fondamentale per l'esecuzione delle istruzioni a partire da un programma risulterà allora chiaro. Esamineremo poi gli altri *componenti* al di fuori del microprocessore che gli forniscono la memoria richiesta e l'agibilità di input/output. Ciò costituirà l'argomento del Capitolo 3. Scendiamo ora nel cuore del microprocessore.

CAPITOLO 2

FUNZIONAMENTO INTERNO DI UN MICROPROCESSORE

OBIETTIVI

Lo scopo fondamentale di questo capitolo è di descrivere l'effettiva architettura e funzionamento di un tipico microprocessore. A tal fine presenteremo le principali alternative disponibili per il progettista che usi microprocessori. Stabiliremo che quasi tutti i microprocessori monolitici hanno essenzialmente un'*architettura comune*. Essi sono sistemi a *bus singolo, imperniati sull'accumulatore*. Esamineremo in dettaglio non solamente l'architettura interna, ma la sequenza effettiva di un dispositivo reale, l'INTEL 8080. Seguiremo l'esecuzione delle istruzioni così come esse provengono dalla memoria entro il *registro istruzioni* (instruction register) interno per essere decodificate ed eseguite. Seguiremo i dati mentre vengono manipolati tra la memoria e i registri interni. Spiegheremo l'uso e il funzionamento dei diversi bus del sistema.

La maggior parte delle limitazioni nella progettazione dei microprocessori è inerente alla struttura LSI. Le limitazioni imposte al progettista di un chip di microprocessore possono essere illustrate meglio mediante un esempio reale. La microfotografia di uno dei primi microprocessori (l'8080 dell'INTEL) compare alla pagina seguente. Essa servirà ad illustrare le caratteristiche e le limitazioni di un tipico dispositivo microprocessore. Il lettore dovrebbe tenere queste limitazioni ben presenti mentre esaminiamo le diverse architetture possibili. Ciò chiarirà perchè la maggior parte dei costruttori abbiano di fatto standardizzato le loro architetture secondo direttrici simili.

Alla fine di questo capitolo dovrebbero essere chiare sia le ragioni delle scelte di architettura (e le limitazioni) sia il reale funzionamento interno di un microprocessore. Sebbene questo capitolo possa apparire molto complicato ad una prima lettura, è decisamente raccomandabile che ogni lettore studi questo capitolo nei dettagli. Mediante una seconda lettura oppure con letture successive tutta la materia dovrebbe diventare chiara e rendere il resto del libro (comparativamente) semplice. Capire questo capitolo è fondamentale per la comprensione del modo in cui un microprocessore funziona e del procedimento della programmazione. Tuttavia non è essenziale per il lettore che non desidera realizzare il proprio sistema. Per questa ragione esso può essere tralasciato durante una prima lettura di questo libro se proprio lo si desiderasse. Gli altri capitoli possono essere letti indipendentemente da questo.

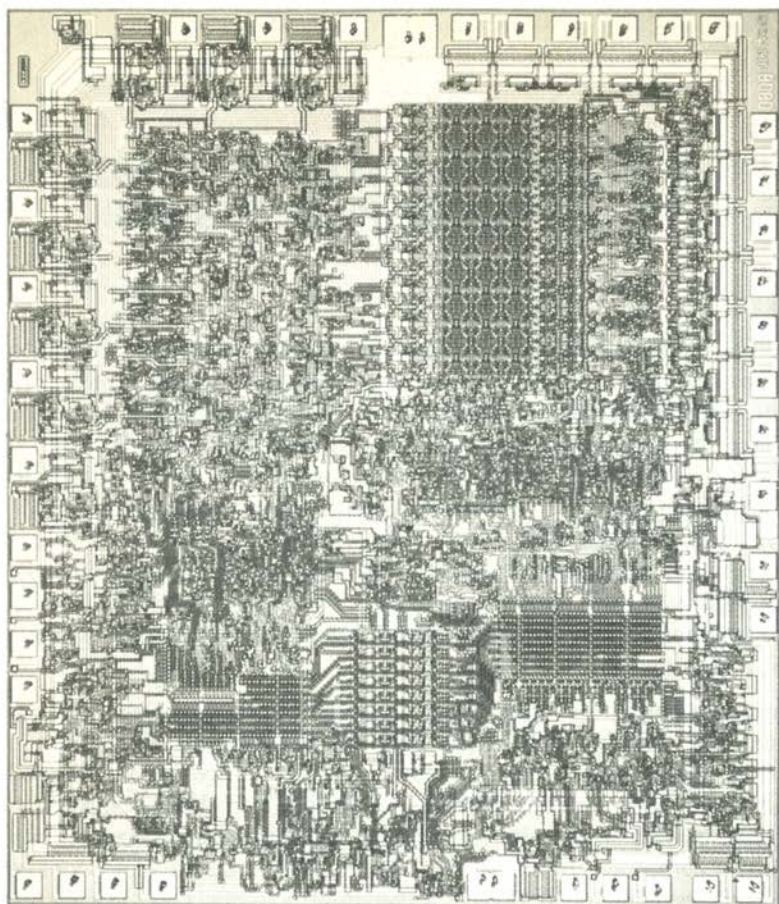



Fig. 2-1: Microfotografia dell'8080.

LE LIMITAZIONI DELL'LSI

Esaminiamo la Fig. 2-1. Sulla fotografia si possono distinguere diverse aree.

Diversi rettangoli si notano facilmente per la loro geometria ripetitiva: infatti il lettore attento può contare 8 celle identiche dalla cima al fondo di questi gruppi di celle. Questi sono *registri a 8 bit.* Queste aree molto geometriche sul chip sono aree di *memoria* di tipo ROM o RAM. La realizzazione di una memoria su di un chip permette il massimo rendimento di utilizzazione dell'area disponibile. Per merito del disegno geometrico e ripetitivo adatto per le memorie, queste aree possono essere molto dense poichè possono essere disegnate automaticamente. In realtà la facilità di realizzazione di una memoria su di un chip è così grande a confronto della

realizzazione di altre funzioni, che i primi progettisti di microprocessori fecero un numero significativo di errori di progetto proprio a causa di questa facilità. I primi microprocessori, ad esempio avevano un eccesso di registri interni. Questi registri interni sono realizzati, infatti, mediante un banco di RAM cioè un insieme di registri di lettura/scrittura realizzati come una RAM. Una piccola RAM richiede solamente una piccolissima area. Come risultato questi microprocessori furono equipaggiati con un gran numero di registri interni, il che sembrava essere un vantaggio. Tuttavia l'indirizzamento di registri multipli implica tipicamente istruzioni più lunghe e perciò un tempo di esecuzione delle istruzioni più lento. Inoltre i primi microprocessori che erano equipaggiati con banchi di registri multipli, avrebbero potuto avere, in linea di principio, un potenziale per una rapida gestione degli *interrupts*. Vedremo che ad essi mancavano altre caratteristiche necessarie per avere dei vantaggi da questa possibilità. In breve un gran numero di registri interni non è un vantaggio per se stesso, a meno che non sia collegato ad altri artifici di sistema che descriveremo.

In particolare, una gestione efficiente degli interrupts mediante banchi di registri multipli, coinvolge anche una duplicazione del *program-counter* e dello *status-word* o *status-register*. Ciò implica, d'altra parte, dei collegamenti speciali sui bus interni. Questi artifizi furono dimenticati su uno dei primi microprocessori a 4 bit di maggior successo commerciale. Questi errori sono stati finalmente corretti, ma solo recentemente.

È probabile che molti progettisti di chip, non appena essi ebbero terminato il loro progetto iniziale, abbiamo rivolto un secondo sguardo al chip vedendo alcune aree vuote ove non era stata realizzata alcuna circuiteria. A questo punto fu probabilmente molto allettante, per loro, aggiungere un certo numero di registri da 4 o 8 bit che potessero essere realizzati in una piccola area del chip.

La massima area di un chip è limitata dalla tecnologia. Dimensioni del chip di 100 mils per 100 mils sono all'incirca la normalità oggi. La dimensione del chip non può essere raddoppiata senza aumentare esponenzialmente i problemi di produzione e perciò aumentare esponenzialmente il costo. Per questa ragione uno dei problemi base è di realizzare tutte le funzioni necessarie, per un microprocessore completo su un'area limitata. Vedremo in uno degli ultimi capitoli che l'aumento dell'area di un chip è una delle preoccupazioni dell'industria oggi e che un lento progresso viene fatto costantemente.

Le aree dall'apparenza disordinata sul chip sono quelle di *random-logic*. I circuiti di *random-logic* devono venir disegnati generalmente a mano e sono soggetti a problemi di affidabilità come la *pattern-sensitivity* (non è possibile prevedere in anticipo tutte le possibili combinazioni di bit che potrebbero comparire nella geometria del chip e che potrebbero attivare un bit accidentalmente). Sebbene esista uno sforzo costante per minimizzare l'area occupata dalla logica random, essa non può utilizzare il chip con un rendimento pari alla memoria.

Infine si può notare in Fig. 2-1 un gran numero di linee di collegamento tra gli elementi del chip, in particolare attorno ai margini. Queste sono linee di comunicazione interne o bus interni del microprocessore. Queste linee sembrano occupare un'area relativamente ampia del chip e di fatto la occupano. Questa è un'area *inattiva* che non può essere utilizzata per realizzare funzioni logiche. Una delle preoccupazioni più importanti dei progettisti è di minimizzare quest'area onde fornire una maggiore complessità funzionale per unità di area del chip. I microprocessori più recenti usano l'area con maggior efficienza; il lettore interessato può riferirsi a microfotografie di microprocessori recenti per verificare questa asserzione.

Infine i rettangoli bianchi sui margini del chip sono i *pads* che serviranno come punti di contatto con il mondo esterno. Un filo d'oro verrà saldato (bonded) ai pad e li collegherà ai pad corrispondenti del package stabilendo il collegamento tra il chip ed il proprio package.

Queste osservazioni di base serviranno ad illustrare le limitazioni che la tecnologia LSI impone all'architettura interna del chip del microprocessore.

I BUS

Un *bus* è un insieme di linee di comunicazione raggruppate per funzione. Tipicamente vengono distinti due tipi di bus: i bus interni ed i bus esterni. In questo capitolo considereremo i bus interni, cioè i bus realizzati sullo stesso chip per interconnettere tutti i suoi elementi logici.

Dal punto di vista sistemistico i bus essenziali del sistema sono i suoi bus esterni. I tre standard di ogni sistema microprocessore sono:

1. il *data-bus*: esso trasmette i dati avanti e indietro tra i vari chip del sistema. Ad esempio esso porterà i dati da un chip di ingresso al microprocessore e riporterà indietro i dati dal microprocessore alla memoria. Nel microprocessore standard esso è un bus bidirezionale ad 8 bit (può essere usato in entrambe le direzioni). È quasi sempre un bus tri-state (logica a 3 livelli), così da poter essere usato in congiunzione con un DMA (direct-memory-access). (Il DMA verrà studiato al Capitolo 3).

2. l'*address-bus*: si diparte dal microprocessore e convoglia gli indirizzi ai dispositivi che possono comunicare con il microprocessore verso l'esterno. Nel caso del nostro microprocessore standard esso è un bus a 16 bit che permette fino a $2^{16} =$ 64K indirizzi esterni. Un'indirizzo portato dall'address-bus normalmente seleziona un dispositivo (di solito un chip) o seleziona un registro entro un dispositivo. L'uso di address bus per indirizzare dispositivi verrà spiegato in dettaglio al Capitolo 5. (Costruzione di un Sistema). L'address bus è sempre usato in unione con il data bus per specificare la sorgente o la destinazione dei dati trasmessi sul data bus. L'address bus ha origine da un registro specializzato del microprocessore, il *program-counter* (PC). Questo collegamento verrà spiegato nei dettagli più avanti in questo capitolo.

3. il *control-bus*: distribuisce i segnali di sincronizzazione al microprocessore prelevandoli dagli altri dispositivi e dal microprocessore agli altri componenti del sistema. I segnali tipici che si propagano sul control bus sono: read, write, interrupt, reset, e riconoscimenti di vario tipo.

Ciascuno dei tre bus creati dal microprocessore possono essere caricati fino ad un carico TTL; la capacità tipica è da 100 a 130 picofarads (approssimativamente da 5 a 7 packages LSI). Nella maggior parte dei sistemi, tranne quelli più piccoli, è perciò necessario aggiungere dei bus drivers sul data bus, l'address bus e qualche volta sul control bus alfine di collegare un numero sufficiente di dispositivi esterni.

Il bus essenziale di un sistema, che definisce la sua architettura, è il suo data bus (o i suoi bus).

Tipicamente l'unico bus che viene mostrato nelle illustrazioni dei sistemi è il data bus (più eventualmente l'address bus). In casi relativamente rari compare il control bus. Internamente un sistema viene definito come avente un'architettura a bus singolo doppio o triplo in funzione del numero dei data bus interni che esso utilizza. Queste architetture saranno ora descritte e valutate. Come risultato di questa valutazione dovrebbe diventare chiaro perchè la maggior parte dei microprocessori standard sono stati realizzati con la medesima architettura di base del bus.

Architettura a bus singolo

Un modo significativo per differenziare le architetture interne dei processori consiste nel contare il numero dei bus usati per comunicare tra i registri e l'unità logica aritmetica. L'architettura piú semplice è quella a bus singolo che è illustrata in Fig. 2-2. In un sistema a bus singolo i dati vengono portati da un registro alla ALU attraverso il bus singolo che compare in cima all'illustrazione.

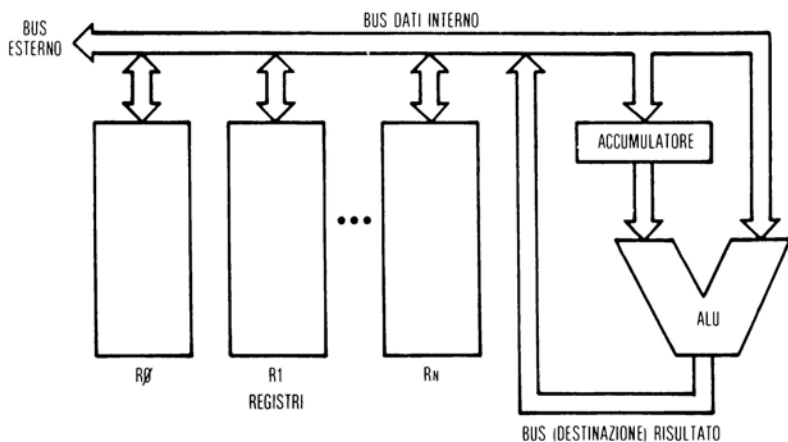



Fig. 2-2: Architettura a bus singolo.

Questo è collegato sia all'ingresso di sinistra che a quello di destra della ALU. I risultati di una operazione eseguita dalla ALU vengono depositati sul medesimo bus per poter essere trasferiti indietro ai registri. Questo bus singolo è *multiplexato* rispetto al tempo.

Una istruzione tipica sarà, ad esempio: R0 = R0 + R1. Questo significa: «ADD (cioè somma) i contenuti di R0 e di R1 e deposita il risultato in R0». Per svolgere questa operazione, i contenuti di R0 verranno letti dal registro R0 trasportati mediante il bus singolo all'ingresso destro della ALU e depositati nell'apposito registro di buffer (o di appoggio) previsto in quell'unità. R1 verrà poi selezionato ed i suoi contenuti verranno letti mediante il bus, quindi propagati verso l'ingresso sinistro della ALU. A questo punto l'ingresso sinistro della ALU è condizionato da R1 e l'ingresso destro è condizionato dal registro di buffer contenente il valore precedente di R0.

L'operazione può quindi essere svolta. L'addizione viene calcolata dalla ALU ed i risultati compaiono sull'uscita della ALU nel lato in basso a destra di Fig. 2-2. I risultati verranno depositati sul bus singolo e propagati indietro ad R0. Ciò significa in pratica che il latch di ingresso (registro di appoggio con possibilità di memorizzazione condizionata) di R0 verrà abilitato, in modo che i dati possano essere scritti su di esso. Ora l'esecuzione dell'istruzione è completa. I risultati della somma sono in R0.

Si dovrebbe notare che i contenuti di R1 non sono stati modificati da questa operazione. Questo è un principio generale: i contenuti di un registro o di qualunque memoria di lettura/scrittura non sono modificati da una operazione di lettura.

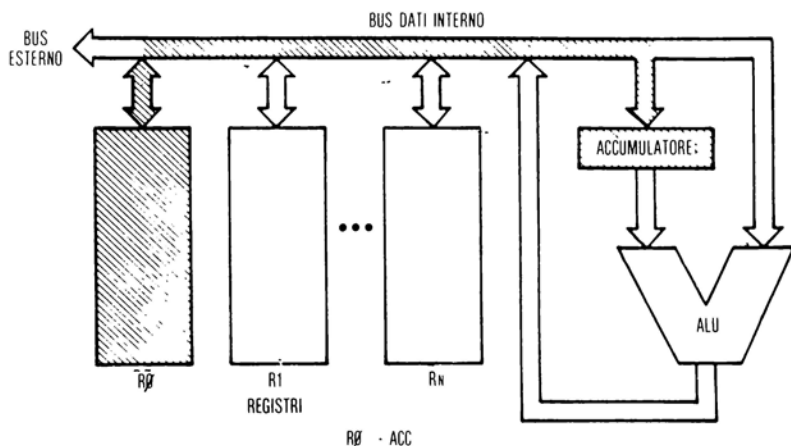



Fig. 2-3: Esecuzione di un'addizione: R0 in ACC.

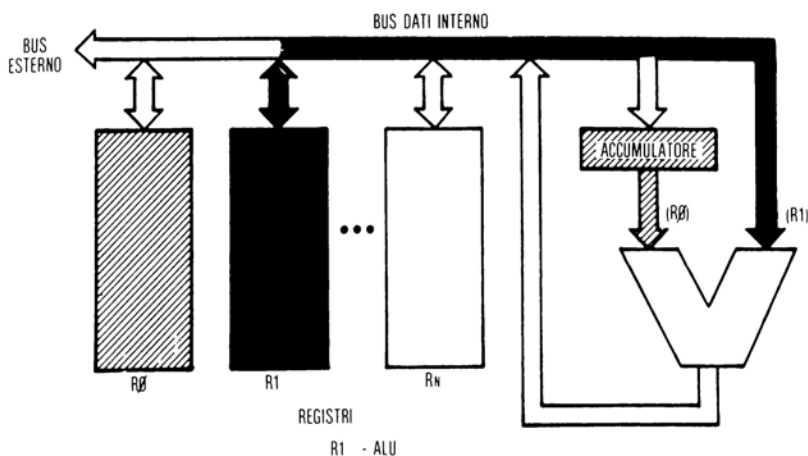



Fig. 2-4: Addizione: il secondo registro (R1) entra nella ALU.

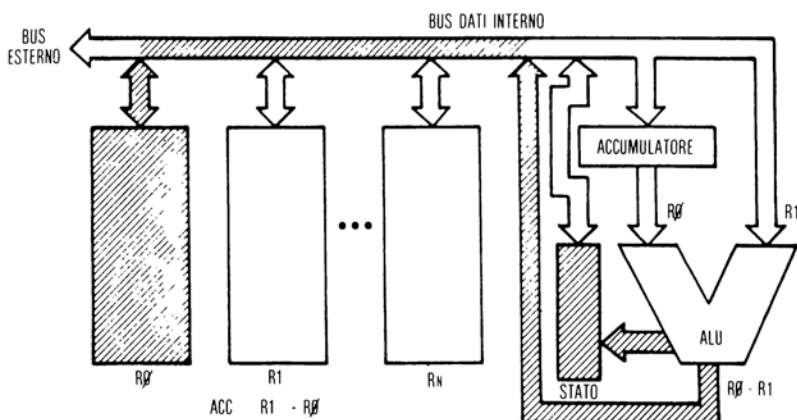



Fig. 2-5: Viene generato il risultato che entra in R0.

Il registro buffer sull'ingresso di destra della ALU era necessario onde *memorizzare* i contenuti di R0, in modo che il bus singolo potesse essere usato ancora per un altro trasferimento. Tuttavia ciò non è ancora sufficiente.

Il problema critico dell'ordine d'arrivo

L'organizzazione semplificata mostrata in Fig. 2-2 non è sufficiente.

DOMANDA: *Qual è il problema delle temporizzazioni?*

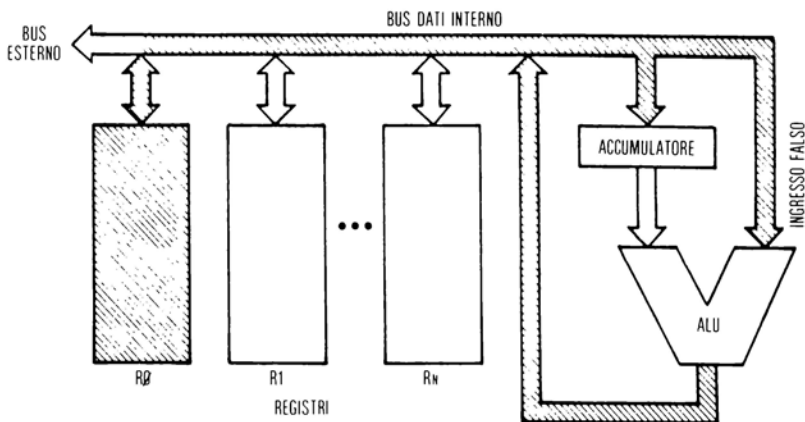



Fig. 2-6: Il problema della corsa critica.

RISPOSTA: *Il problema sorge dal fatto che il risultato che sarà propagato in uscita dall'ALU verrà depositato indietro sul bus singolo. Esso non si propagherà solo in direzione di R0, ma lungo tutto il bus. In particolare esso ricondizionerà l'ingresso sinistro dell'ALU cambiando pochi microsecondi piú tardi il risultato uscente da essa. Questa è una corsa critica (critical race). L'uscita dell'ALU deve venire isolata dal proprio ingresso.*

Per isolare l'ingresso della ALU dall'uscita sono possibili diverse soluzioni. Bisogna usare un registro di buffer. Il registro di buffer potrebbe essere posto sull'uscita dell'ALU o sui suoi ingressi. Essa è generalmente posta sull'ingresso dell'ALU. Qui

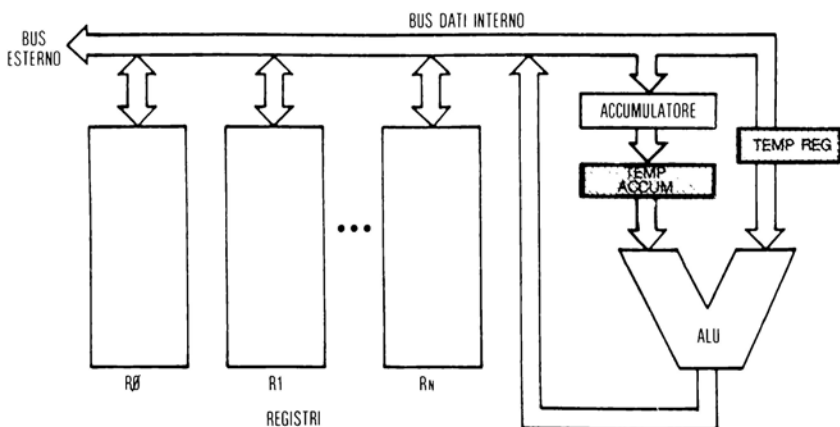



Fig. 2-7: Sono richiesti due buffers.

essa potrebbe essere posta sul suo ingresso sinistro. L'isolamento mediante buffer del sistema è ora sufficiente per un corretto funzionamento. Verrà mostrato piú avanti in questo capitolo che se il registro di destra, che compare in questa illustrazione, deve essere usato come accumulatore (permettendo l'uso di istruzioni lunghe 1-byte) allora anche l'accumulatore richiederà un buffer.

Vantaggi e Svantaggi

Lo *svantaggio* di un sistema a bus singolo è il suo funzionamento lento: sono necessari tre trasferimenti: dal registro R1, quindi dal registro R0 e indietro dall'ALU al registro del risultato. Il bus è multiplexato. Per migliorare la velocità di esecuzione di un processore è desiderabile usare bus multipli. Si possono usare due o tre bus. Descriveremo qui sotto una architettura a bus triplo.

Il *vantaggio* fondamentale di un sistema a bus singolo dovrebbe essere altrettanto chiaro; è l'architettura che richiede la minore area di bus. Su un chip essa fa risparmiare spazio. Questa è la considerazione critica, oggi, per la realizzazione di un microprocessore. Come risultato la maggior parte dei chip microprocessori oggi usa architetture a bus singolo. I progetti ad alta funzionalità, come quelli che utilizzano il bit-slice, non sono limitati da questa considerazione e dispongono almeno di tre bus separati.

ARCHITETTURA A BUS DOPPIO E TRIPLO

Un sistema a bus doppio potrebbe usare, ad esempio, un unico bus d'ingresso che collegherebbe entrambi gli ingressi della ALU e un bus separato per il risultato

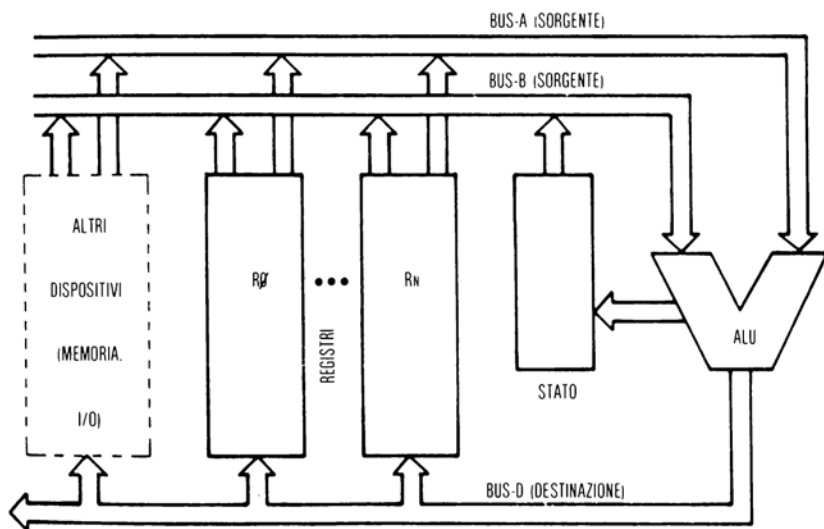



Fig. 2-8: Architettura a bus triplo.

(«D-bus») connesso dalla ALU indietro verso i registri. Ad esempio molti prodotti della National Semiconductor usano questa architettura per aumentare la velocità.

L'architettura di un sistema a *bus triplo* fornisce la massima funzionalità. Esso compare in Fig. 2-8. Sono stati previsti due bus di ingresso, il «bus –A» e il «bus-B». Il bus-A è collegato all'ingresso destro della ALU; il bus-B è collegato all'ingresso sinistro della ALU. Entrambi gli ingressi della ALU possono ora venir selezionati simultaneamente. Inoltre essi non necessitano di alcun buffer sebbene i buffer possano essere inseriti per altre ragioni. In piú i risultati possono essere prelevati sul bus-D indipendentemente dai due bus sorgente. Se il risultato deve essere scritto

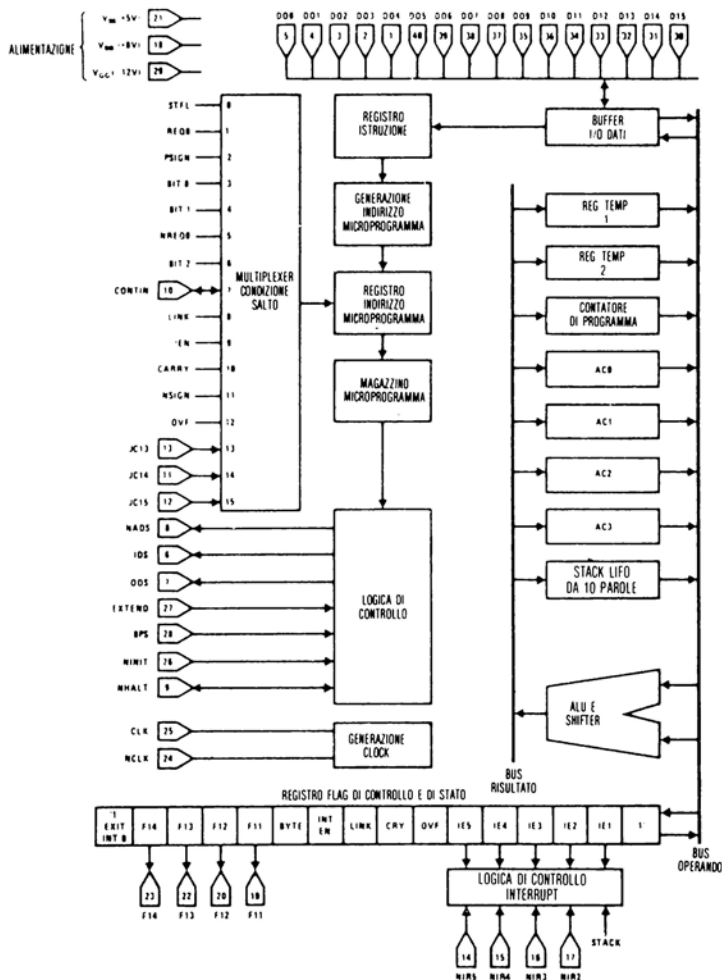



Fig. 2-9: Il microprocessore Pace.

indietro su uno dei registri sorgente, diventa implicita una funzione di buffer. Questa funzione di buffer deve essere eseguita sul bus-D o direttamente entro i registri come avviene di solito.

DOMANDA: *In Fig. 2-9 appare l'architettura del microprocessore Pace, il primo microprocessore a 16-bit (introdotto dalla National Semiconductor). Quanti bus ha? (Ignorate la metà di sinistra e il fondo dell'illustrazione per questo esempio. I bus interni e i registri compaiono nella metà di destra della figura).*

ARCHITETTURA STANDARD DI UN MICROPROCESSORE

La grande maggioranza dei chip microprocessori oggi sul mercato sono realizzati con la medesima architettura. Questa architettura «standard» sarà qui descritta. Si tratta di un'architettura a singolo bus che utilizza l'area del chip con il massimo rendimento. Questo progetto standard appare in Fig. 2-10. I moduli di questo microprocessore standard saranno ora descritti in dettaglio, da sinistra a destra.

Il *blocco di controllo* (control box) sulla destra rappresenta la unità di controllo (control-unit) che sincronizza il funzionamento dell'intero sistema. Il suo ruolo sarà chiarito nel seguito di questo capitolo. Al momento non è essenziale.

La *ALU* esegue operazioni aritmetiche e logiche. Virtualmente tutti i microprocessori utilizzano una architettura a bus singolo imperniata su un accumulatore.

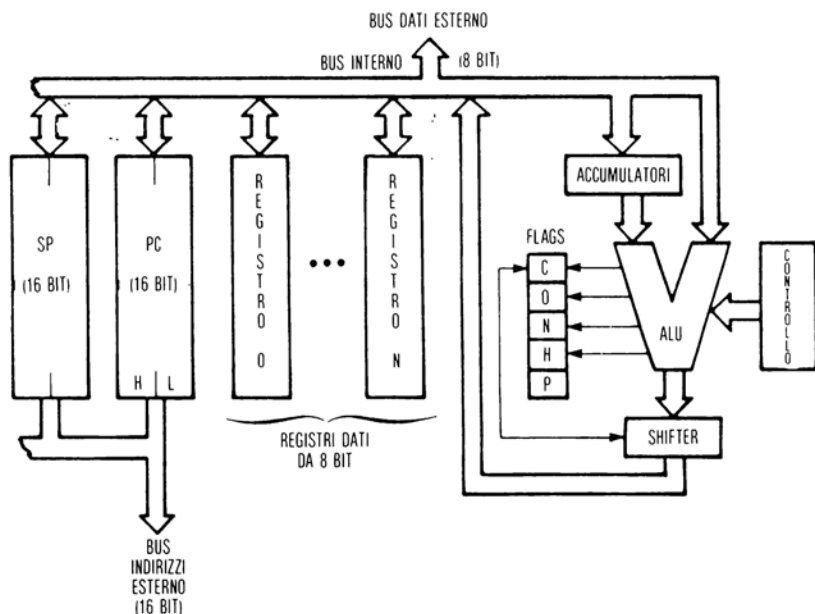



Fig. 2-10: Architettura standard del microprocessore.

Uno degli ingressi della ALU, quello di sinistra in questo esempio, è munito di un registro speciale. Esso è l'accumulatore. (Si possono inserire diversi accumulatori). Entro la medesima istruzione ci si può riferire all'accumulatore sia come input, sia come output. Il suo vantaggio consiste nell'introdurre un livello in piú di separazione per accedere all'ALU, un rallentamento ogni volta che dei registri esterni devono trasferire i loro contenuti attraverso di esso.

La ALU deve anche fornire la possibilità di eseguire degli *shift* e dei comandi di *rotate* (rotazione).

Un'operazione di shift consiste nello spostamento dei contenuti di un byte di una posizione verso sinistra o verso destra. Ciò è illustrato in Fig. 2-11. Ciascun bit è stato spostato di una posizione verso sinistra.

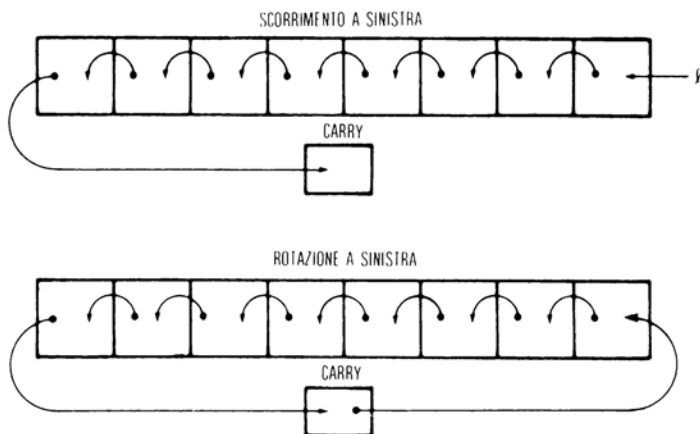



Fig. 2-11: Scorrimento e rotazione.

A questo punto sorgono spontaneamente due domande: Quale bit entrerà dalla destra? E che cosa succede al bit che viene «scaricato» sulla sinistra?

In uno shift regolare il bit entrante dalla destra è uno «0». Il bit scaricato sulla sinistra viene catturato in un bit del registro speciale di stato, il bit di «carry» (o riporto). Esso viene immagazzinato in quella posizione dove può essere «testato» (n.d.t.: brutto neologismo che indica un'azione di verifica per constatare se detto bit è a 0 o a 1). Esso può persino essere trasferito indietro all'ingresso di destra dell'accumulatore. Quest'ultima è chiamata un'operazione di *rotate* ed è illustrata in Fig. 2-11. In un'operazione di rotate ciascun bit viene fatto scorrere a sinistra di una posizione. Il bit che rientra dalla destra del registro è il valore precedente del bit di carry. Alla fine dello shift il bit più a sinistra (left most bit) che viene scaricato determinerà il nuovo valore del bit di carry. Matematicamente questa è una rotazione a nove bit.

In ciascun microprocessore i dettagli possono variare. In particolare per l'efficienza delle operazioni aritmetiche è necessario provvedere diversi shift e rotate come la possibilità del «sign-extend», che ripete il valore del left-most bit durante un'operazione di shift a destra.

Tali possibilità mancano su alcuni microprocessori. Una *reale rotazione* consisterebbe nello scrivere indietro il bit piú a sinistra direttamente e contemporaneamente a quello entrante sulla destra. Ciò normalmente non viene fatto nel mondo dei microprocessori.

La ALU deve essere equipaggiata con uno shifter. Lo shifter può essere sull'uscita della ALU come illustrato in Fig. 2-10 oppure può essere sull'ingresso dell'accumulatore.

Sulla sinistra della ALU compaiono i *flag* o *status-register*. Il loro ruolo è di memorizzare condizioni eccezionali nell'ambito della ALU. I contenuti del registro dei flag può essere verificato da istruzioni specializzate o può essere letto sul data bus interno. Le istruzioni di *branch condizionale* determineranno l'esecuzione di un nuovo programma in funzione del valore di uno di questi bit.

Esaminiamo qui il ruolo dei bit di stato nel microprocessore. Analizzeremo il registro dei flag da cima a fondo:

C (carry)

Il bit di carry svolge due funzioni differenti ed indipendenti. Esse sono riunite su un unico bit mentre su un processore piú grande sarebbero separate.

1. *Il carry memorizza il carry aritmetico*, cioè il nono bit che può essere generato durante un'operazione aritmetica. Esso è il riporto (overflow) del risultato di otto bit. Ad esempio se si addizionano i seguenti due numeri:

il risultato genera un riporto (carry) cioè un nono bit. L'1 generato da questa somma verrà memorizzato dalla ALU nel bit C dove può essere testato. Alcune istruzioni speciali come «ADD con riporto» sommeranno automaticamente il riporto al risultato della somma successiva. Altrimenti il programmatore potrà eseguire un test usando una istruzione di branch condizionale per determinare se deve essere intrapresa qualche azione.

2. *Il carry viene usato come spill-out* durante le operazioni di shift e rotate. Quando usato in questo modo il carry si comporta ancora come nono bit del registro, il

che giustifica l'unione di queste due funzioni in un medesimo bit. L'altra ragione di questa unione è che essa facilita e aumenta la velocità delle operazioni aritmetiche di moltiplicazione e di divisione.

Ciò servirà anche a chiarire una osservazione che fu fatta precedentemente. Sfortunatamente nella maggior parte dei microprocessori non è possibile testare qualsiasi bit entro un dato registro e persino nemmeno entro l'accumulatore. L'unico bit che può essere testato direttamente entro l'accumulatore è il bit sette che è effettivamente il flag N dello status register. Qualora un altro bit dell'accumulatore debba essere testato il programmatore dovrà eseguire una serie di shift. Uno shift sposterà uno dei bit dell'accumulatore entro il bit di carry dove verrà testato.

O (overflow)

L'overflow denota il fatto che un riporto aritmetico entro la parola ha modificato il valore del bit piú significativo determinando un errore nel segno nel caso di una annotazione complemento a 2 che verrà spiegata nel capitolo riguardante la Programmazione. Il bit 7 in complemento a 2 (l'MSB) indica il segno del numero. «1» è negativo, «0» è positivo. Ogni qualvolta due numeri in complemento a 2 vengono sommati il riporto generato durante un'addizione o sottrazione potrebbe andare in overflow (straripare) nel bit del segno. Quando ciò avviene un numero negativo potrebbe tramutarsi in un numero positivo. Il bit di overflow indica che ciò è avvenuto. Matematicamente l'overflow è l'OR esclusivo del bit di carry proveniente dal bit 7 con il riporto generato dal bit 6 entro il bit 7. L'overflow sarà usato normalmente solo quando si eseguano operazioni aritmetiche in complemento a 2.

N (negative)

Il bit N è direttamente collegato alla posizione di bit 7 del risultato. Nella notazione complemento a 2 il bit 7, l'MSB, indica il segno del numero. Un «1» in questa posizione indica un numero negativo, di qui il nome di bit di stato. È stato precisato che il bit 7 è l'unico bit dell'accumulatore che può essere prontamente testato, per merito della disponibilità di N entro lo status register. Ecco perchè la posizione preferita per memorizzare lo stato in qualsiasi latch di input/output o registro è la posizione del bit 7. Nello svolgere operazioni aritmetiche il bit N verrà usato per determinare se un numero o un risultato è positivo o negativo.

H o AC (half-carry)

Questo bit è usato durante le operazioni *BCD.* BCD sta a significare binary-coded-decimal. Il binary-coded-decimal verrà spiegato nel capitolo riguardante la Programmazione. Il BCD utilizza un codice a 4 bit per rappresentare ciascuna cifra decimale. Per ragioni di compattezza dei dati una parola standard di 8 bit conterrà due codificazioni BCD l'uno fianco all'altra. Quando si eseguono operazioni aritmetiche su bytes un'addizione potrebbe generare un riporto dal bit 3 entro il bit 4 cioè dalla prima cifra BCD entro la seconda cifra BCD. Questo riporto normalmente non è desiderabile e deve essere indicato. Questo è il ruolo del bit H. Il bit H

è il riporto dal bit 3 al bit 4. L'annotazione BCD è usata nelle applicazioni amministrative che richiedono risultati esatti senza errore di arrotondamento come nel caso della annotazione binaria. Lo svantaggio del BCD sta nell'occupazione di spazio e nell'utilizzazione inefficiente dello spazio di memoria nonchè nella lentezza di calcolo aritmetico.

Z (zero)

Il bit Z viene settato (gli viene attribuito un valore 1) quando il risultato di un'operazione è 0. Viene usato dalle istruzioni aritmetiche per determinare se un risultato è 0 ed anche per operazioni logiche come quelle di comparazione (compare). Qualora il risultato di un paragone dia esito positivo il bit Z verrà settato a l. Ciò perché l'operazione di compare realizza di fatto un XOR logico tra la parola da testare e la configurazione alla quale deve essere paragonata.

Il bit Z viene frequentemente testato dalle istruzioni di input e di output per determinare se i contenuti di un qualche bit entro un registro sono cambiati. È possibile testare se una configurazione di bit è cambiata oppure no semplicemente eseguendo un XOR sui valori del registro con i suoi precedenti valori. Se nessun bit è cambiato il risultato è 0 e ciò verrà indicato dal bit Z. Se un qualsiasi bit è cambiato il risultato del XOR non sarà 0 e ciò verrà indicato dal bit Z.

P (parity)

Questo bit non viene normalmente inserito nella maggior parte dei microprocessori, ma è presente in uno dei primi (l'8080). La parità è utilizzata per indicare il fatto che i dati sono stati trasmessi correttamente (oppure incorrettamente). Il principio della parità consiste nel contare il numero degli 1 presenti negli 8 bit. Uno schema a parità pari completerà il numero di 1 parola a 7 bit aggiungendo uno 0 o 1 in modo che il numero totale risulti pari. Di converso uno schema a parità dispari agirà sull'ottavo bit in modo che il totale degli 1 contenuti negli 8 bit sia dispari.

Il bit di parità viene usato per indicare se il conteggio di parità è corretto o meno. Assumendo ad esempio una parity pari il bit di parità verrà settato ogni volta che il numero di bit entro la parola non risulti pari, indicando un errore di parità.

Poichè la parity è usata essenzialmente per le comunicazioni la possibilità di eseguire test di parità non è normalmente fornita entro i microprocessori. Essa viene normalmente realizzata nella UART che verrà descritta al prossimo capitolo. I dati che arrivano nella MPU vengono considerati corretti. La maggior parte dei microprocessori esistenti trascurano di verificare la parità e sono sprovvisti della relativa circuiteria.

Altri Bit di Stato

Altri statut bit potrebbero essere inseriti nel registro dei flag.. in particolare potrebbe essere stato previsto un bit di interrupt. La sua funzione è normalmente

quella di abilitazione dell'interrupt (*interrupt-enable*). Qualora esso venga settato gli interrupt dall'esterno verranno accettati. Qualora non sia settato («0») gli interrupt esterni saranno inibiti; essi si diranno mascherati (*masked*). Questo procedimento verrà descritto in dettaglio al Capitolo 3 nel paragrafo sugli interrupt.

Talora altre informazioni di stato possono essere presenti in questo registro raggruppando in quest'altro flip-flop interni che memorizzino eventi interni del processore. Ecco perché questo registro è usualmente chiamato anche *PSW* o Program-Status-Word. Esso memorizza l'informazione completa di stato del processore per quel programma.

Modifica dei Flag

La maggior parte delle istruzioni eseguite dal processore modificherà qualcuno o tutti i flag. È importante riferirsi sempre alla tabella fornita dal costruttore che elenca quali bit verranno modificate dalle istruzioni. Ciò è essenziale per capire il modo in cui un programma viene eseguito.

I Registri

Torniamo ora alla Fig. 2-5 e spostiamoci verso sinistra. Sulla sinistra dell'illustrazione compaiono i registri del microprocessore. Si possono distinguere i registri di uso generale (*general-purpose register*) e i registri di indirizzo (*address register*).

I General-Purpose Register

I general-purpose register devono venir previsti per permettere alla ALU di manipolare i dati ad alta velocità. A causa di restrizioni sul numero dei bit che è ragionevole prevedere in una istruzione il numero dei registri (direttamente indirizzabile) è generalmente limitato a meno di otto. Ciascuno di questi registri è un'insieme di otto flip-flop collegati al data bus bidirezionale interno. Questi otto bit possono essere trasferiti simultaneamente da o verso il data bus. La realizzazione di questi registri con flip-flop MOS fornisce il piú veloce livello disponibile di memoria e il loro contenuto può essere richiamato entro decine di nanosecondi.

I registri interni sono usualmente contrassegnati da 0 a n. Il ruolo di questi registri non è definito in anticipo: essi sono detti infatti «general purpose» o di impiego generale. Essi possono contenere dati utilizzati dal programma.

Questi registri general-purpose verranno normalmente usati per memorizzare dati di 8 bit. Su qualche microprocessore esistono possibilità di manipolare *due* di questi registri per volta. Essi sono chiamati in tal caso registri a coppie «register-pairs». Ciò facilita la memorizzazione di quantità a 16 bit, siano essi dati oppure indirizzi.

Gli Address Register

Gli address register (o registri d'indirizzamento) sono registri a 16 bit dedicati a contenere gli indirizzi. Essi sono anche chiamati spesso *data-counter* oppure *pointer*. Essi sono registri accoppiati cioè due registri da 8 bit. La loro caratteristica es-

senziale consiste nell'essere collegati al bus degli indirizzi. Gli address-register creano l'address-bus. L'address-bus compare sulla sinistra e in fondo all'illustrazione in Fig. 2-12.

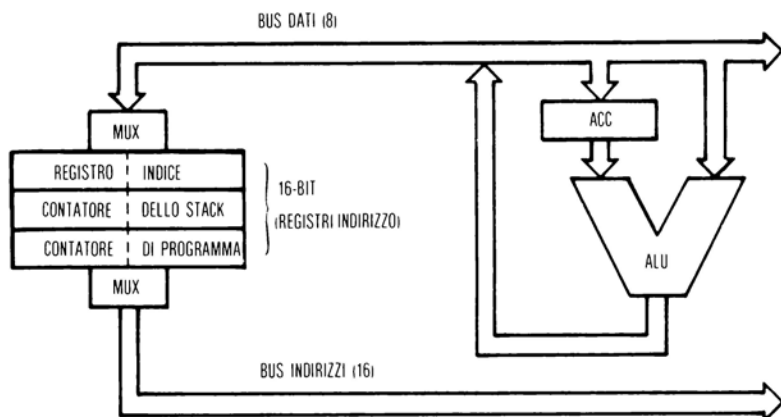



Fig. 2-12: I registri di indirizzo a 16 bit originano il bus indirizzi.

L'unico modo per modificare i contenuti di questi registri a 16 bit è mediante il data bus. Per trasferire 16 bit lungo il data bus saranno necessari due trasferimenti. Al fine di differenziare la metà inferiore dalla metà superiore di ciascun registro, essi sono generalmente etichettati con L (low, cioè inferiore) e con H (high, cioè superiore), denotando rispettivamente i bit da 0 a 7 e da 8 a 15. Questa etichetta è usata ogni volta che sia necessario differenziare una delle due porzioni di questi registri. Nella maggior parte dei microprocessori sono presenti almeno due address register.

-PC-Program Counter

Il Program counter deve essere presente in qualsiasi processore. Esso contiene l'indirizzo della successiva istruzione da eseguire. La presenza di un program counter (o contatore di programma) è indispensabile e fondamentale per programmare l'esecuzione. Il meccanismo di esecuzione del programma e la sequenzializzazione automatica realizzata con il program counter verranno descritti nel prossimo capitolo. In breve l'esecuzione di un programma è normalmente svolta in modo sequenziale. Per poter accedere all'istruzione successiva, è necessario trasferirla dalla memoria entro il microprocessore. I contenuti del PC verranno depositati sull'address bus e propagati verso la memoria. La memoria leggerà allora i contenuti specificati da questo indirizzo ed invierà di ritorno alla MPU la parola corrispondente. Questa è l'istruzione. In pochi microprocessori eccezionali come l'F8 a due chip, non esiste alcun PC sul microprocessore. Ciò non significa che il sistema non sia fornito

di program counter. In questo caso il PC è stato realizzato direttamente sul chip di memoria per ragioni di efficienza di questa architettura.

-SP-Stack Pointer

Lo *stack* (pila o catasta di registri) non è stato ancora definito e verrà descritto al prossimo paragrafo. Nella maggior parte dei microprocessori di impiego generale lo stack viene realizzato in «software» cioè utilizzando la memoria. Onde tener traccia della posizione di testa di questo stack entro la memoria un registro a 16 bit viene dedicato alla funzione di pointer (puntatore). Lo stack pointer SP contiene l'indirizzo di testa dello stack entro la memoria. Verrà dimostrato che lo stack è indispensabile per gli interrupts e per le soubroutines.

-IX-Index Register

L'indicizzazione (indexing) è un'opportunità di indirizzamento della memoria che non è sempre disponibile nei microprocessori. Le varie tecniche di indirizzamento della memoria saranno descritte nel capitolo Programmazione. L'indicizzazione è un'opportunità di accedere a blocchi di dati nella memoria con una singola istruzione. Un registro indice conterrà tipicamente una grandezza di spostamento che verrà automaticamente sommata ad una base oppure potrebbe contenere una base che verrebbe sommata ad una grandezza di spostamento o displacement).
L'indicizzazione è usata per accedere a qualsiasi parola entro un blocco di istruzioni. Il Motorola 6800 o il Signetics 2650 sono muniti di un registro IX. L'8080 non ha un registro IX. Tuttavia alcuni registri general purpose dello 8080 possono essere usati accoppiati per memorizzare un numero usato come indice. Un paragone tra l'8080 ed il 6800 è presentato al Capitolo 4.

Alcuni altri microprocessori, come l'SC/MP della National utilizzano registri a 16 bit general purpose che possono essere usati come pointers d'impiego generale. Essi possono essere usati come indici o stack pointers usando la possibilità di auto indicizzazione.
Infine esistono alcune strutture specializzate di registri che possono essere disponibili sul chip del microprocessore.

Lo Scratchpad

Uno *scratchpad* è un insieme di registri interni d'impiego generale. Uno scratchpad è di fatto una RAM interna. Ciò implica che un indirizzo deve essere specificato prima che uno qualsiasi di questi registri possa essere utilizzato. Il nome è derivato di fatto che questi registri interni (eventualmente 64 o anche più unità) sono usati essenzialmente per immagazzinare informazioni temporanee in un mezzo veloce. Un tale scratchpad sarebbe normalmente collegato sia al data-bus, sia all'address-bus. La differenza tra i registri diretti e quelli dello scratchpad sta nel loro diverso indirizzamento tipo memoria, cioè istruzioni da 2 o 3 byte.

L'utilizzo di uno scratchpad è più veloce che un accesso in memoria, ma è più lento che utilizzando i registri general - purpose.

Lo Stack

Uno *stack* è una struttura «last-in-first-out» (LIFO). Essa è una struttura *cronologica,* che accumula gli eventi (o simboli) nell'ordine nel quale essi vengono depositati. Il simbolo «più vecchio» si trova in fondo allo stack e quello «più recente» in cima. Uno stack può essere paragonato ad una pila di piatti in un bar.

I piatti sono impilati in una cavità circolare munita di una molla. I nuovi piatti sono posti in cima alla pila. Ogni volta che un piatto viene tolto esso viene sempre tolto dalla cima: l'ultimo elemento entrato (last-in) è il primo ad uscire (first-out). L'accesso normale allo stack è dalla testa. Uno stack è gestito da due istruzioni: PUSH e POP (o PULL). Una operazione di PUSH depositerà i contenuti di un registro nello stack. Un'operazione di POP asporterà la cima dello stack e la depositerà entro un registro. Gli stack sono necessari per fornire i livelli di interrupt e di subroutine. Essi possono essere realizzati in due modi fondamentali: hardware e software.

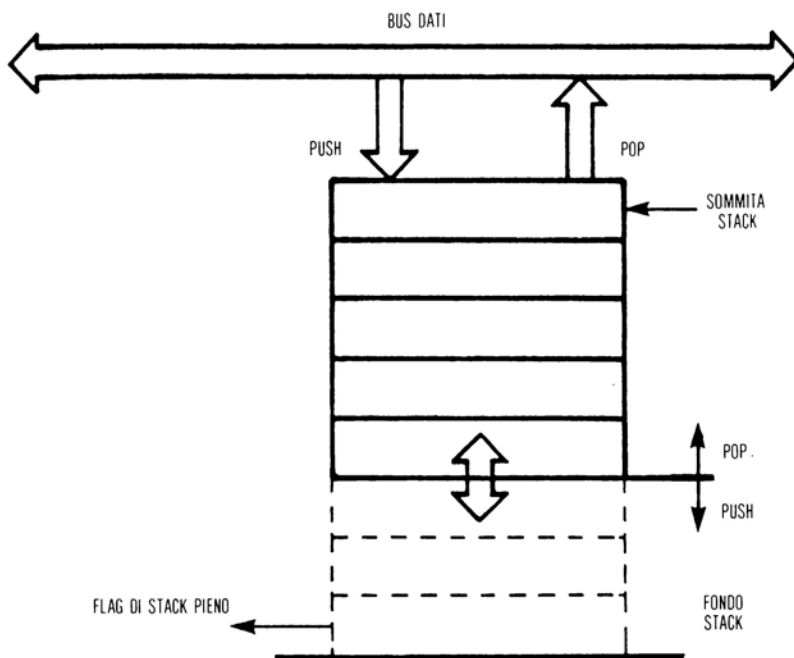



Fig. 2-13: Allo stack si accede dall'alto.

Uno *stack-hardware* è realizzato mediante un insieme di registri interni direttamente sul chip della MPU. Alla operazione di stack sono dedicati allora N registri. Il vantaggio di questa soluzione è l'alta velocità. Lo svantaggio consiste nella profondità limitata dello stock. Ogni volta che gli n registri sono pieni lo stack interno è pieno. Per poter proseguire ad utilizzarlo è necessario ricopiare tutti gli n registri entro la memoria. In più ciò dà adito ad una nuova difficoltà: si deve sapere quando lo stack è pieno o vuoto. La maggior parte dei costruttori semplicemente dimentica questo dettaglio e la parte dei primi microprocessori non metteva a disposizione un flag di «stack pieno» o di «stack vuoto». In tali microprocessori si poteva addirittura continuare ad infilare dati nello stack ed essi sarebbero caduti attraverso

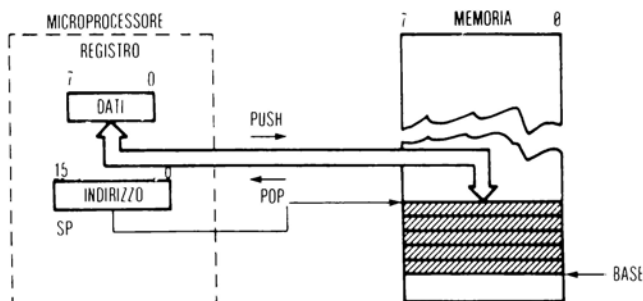



Fig. 2-14: Le due istruzioni di manipolazione dello stack.

l'ultima parola senza che il programmatore lo potesse neppure sapere. Naturalmente questo sarebbe stato un errore di programmazione. Il programmatore avrebbe dovuto stare più attento. In pratica un flag risolve il problema. Di converso è possibile continuare ad estrarre elementi da uno stack per l'eternità. Per questa ragione è consigliabile avere un indicatore di «stack vuoto». L'idea di uno stack hardware fu particolarmente attraente per i costruttori dei primi microprocessori. La ragione di ciò derivava dal fatto che essi non avevano la sensazione che gli interrupts fossero una possibilità molto richiesta e pensavano che i livelli di interrupt, in ogni caso, non sarebbero stati molti in considerazione della lentezza di esecuzione di questi primi microprocessori. In più è particolarmente conveniente usare qualsiasi area «inutilizzata» sul die per realizzare registri addizionali. È stato chiarito che i registri costituiscono l'utilizzazione a più alto rendimento di qualsiasi area del chip disponibile. Fu perciò particolarmente attraente fornire un gran numero di registri interni come ad esempio disponibilità di stack sul chip. Questo fu di fatto l'approccio scelto, con frequenti spiacevoli risultati.

L'alternativa è uno *stack software.* Al fine di disporre di uno spazio ad accrescimento «illimitato» per lo stack, lo stack viene realizzato nella memoria a lettura/scrittura del sistema cioé nella RAM. La base dello stack viene scelta dal programmatore. La cima dello stack viene gestita automaticamente entro il registro

SP. Ogni volta che viene eseguita una operazione di PUSH l'SP viene incrementato o decrementato in funzione della convenzione utilizzata cioè in dipendenza dal fatto che la memoria «cresca» o «si riduca» dal fondo verso la cima. Tutte le volte che viene eseguito un POP, lo stack pointer viene aggiornato automaticamente in modo simile. In pratica l'SP indica usualmente la parola appena *al di sopra* dell'ultimo elemento dello stack. Indica perciò la prima parola disponibile dello stack. Lo scopo è di permettere una operazione di PUSH più veloce possibile al contrario di quanto avverrà per il POP (ciò è essenziale al momento di un interrupt o per catturare un blocco di caratteri). Con questa convenzione, lo stack pointer può essere usato direttamente, senza dover attendere un incremento, quando diventa necessario salva-

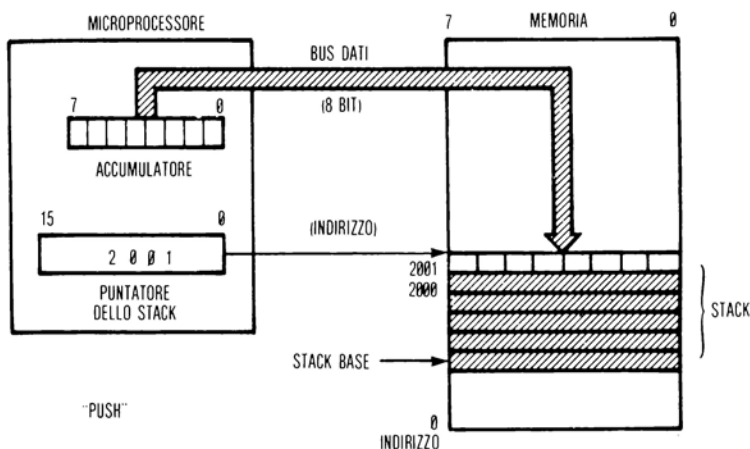



Fig. 2-15: Un'introduzione (PUSH) nello stack.

Fig. 2-16: La MPU dopo un'istruzione PUSH.

re rapidamente alcune parole nello stack, come avviene nel caso degli interrupts. Descriveremo qui di seguito una operazione di PUSH A ed una di POP A.

L'operazione di PUSH è illustrata in Fig. 2-15. I contenuti dell'accumulatore (A) vengono trasferiti in testa allo stack.

L'indirizzo della prima parola disponibile nello stack (2001) è contenuto inizialmente in SP. Dopo l'esecuzione di PUSH, l'SP è stato automaticamente incrementato al valore 2002 (vedi fig. 2-16) e punta verso la nuova «prima locazione disponibile».

Di converso, un'istruzione POP A preleva l'elemento di testa dello stack (vedi Fig. 2-17) e lo carica entro l'accumulatore. Il valore iniziale di SP era 2002. Esso viene automaticamente decrementato a 2001 prima di eseguire il fetch dalla memoria.

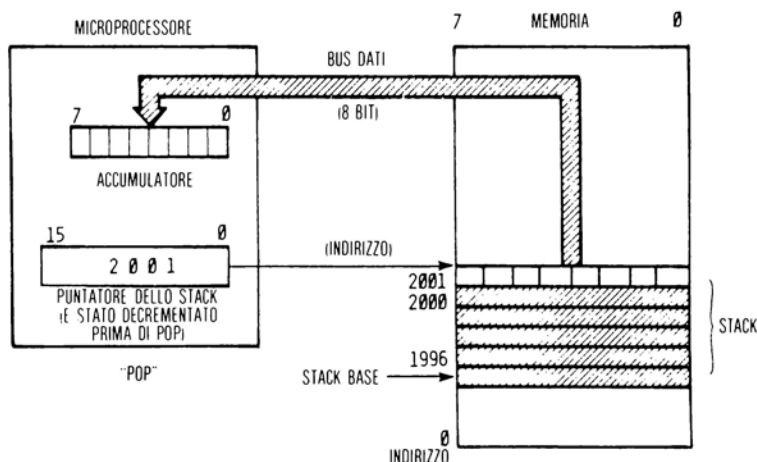



Fig. 2-17: Un prelievo dallo stack (POP).

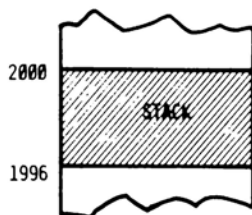



Fig. 2-18: La memoria dopo l'istruzione (POP).

L'aspetto finale dello stack appare in Fig. 2-18.

Sono possibili ancora altre realizzazioni che siano combinazioni di metodi hardware e software. Esse sono realizzate raramente nei microprocessori e non saranno descritte qui.

L'Address - Bus

Il nostro microprocessore standard ora ha due bus interni. Il data-bus bidirezionale si estende verso l'esterno mediante un buffer (separatore) tri-state. I pin del data bus sono normalmente contrassegnati da D0 a D7. L'address-bus è creato dai registri di indirizzamento a 16 bit. I suoi pin sono normalmente contrassegnati da A0 ad A15. La dimensione dell'address-bus è per convenzione di 16 bit, per la facilità di trattare multipli di 8 bit.

A questo punto sono stati descritti tutti gli usuali elementi funzionali del nostro microprocessore standard a 8 bit. I chip più recenti possono incorporare altre possibilità nell'ambito del medesimo chip, come un circuito di clock, un circuito di temporizzazione ed altri. Queste funzioni addizionali sono esterne al chip microprocessore nei progetti tradizionali e verranno descritte in dettaglio al Capitolo 3. Esamineremo ora il ruolo relativo dei registri e dei bus: Seguiremo a tal fine l'esecuzione di una istruzione entro il chip.

Esecuzione di una Istruzione

L'esecuzione di una istruzione è illustrata in Fig. 2-19. Il chip MPU compare sulla sinistra e un chip di ROM contenente il programma compare sulla destra. La sequenzializzazione fondamentale è la seguente:

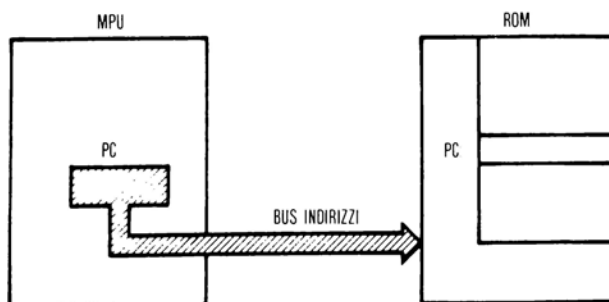



Fig. 2-19: L'istruzione è prelevata all'indirizzo indicato da PC.

Ciascuna istruzione viene eseguita come una sequenza di tre fasi: fetch - decodifica - esecuzione. L'istruzione verrà dapprima *fetched* (prelevata) dalla memoria e trasferita entro il microprocessore in uno speciale registro della control unit chiamato IR, o *instruction-register*. Una volta depositata nell'IR, essa verrà *decodificata* da un decoder. Infine essa verrà *executed* (eseguita). L'appropriata sequenza

di segnali verrà generata dalla control unit e ciò condurrà agli appropriati trasferimenti di dati e operazioni.

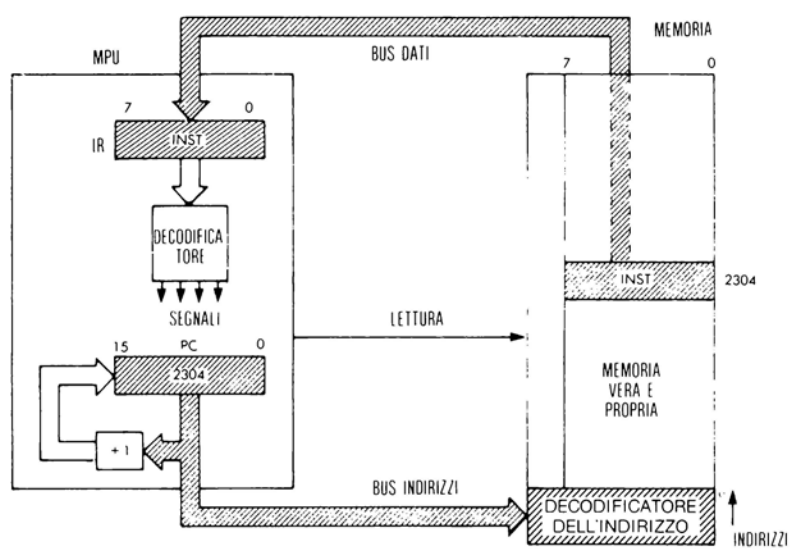



Fig. 2-20: L'istruzione è letta dalla memoria in IR.

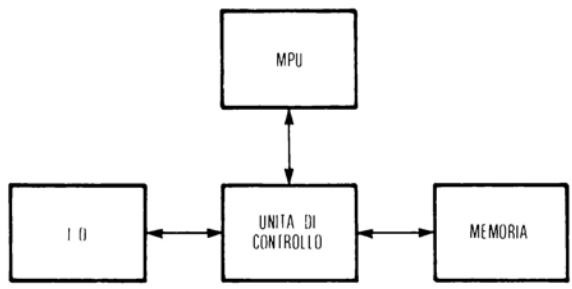



Fig. 2-21: L'unità di controllo sincronizza il sistema.

Torniamo indietro alla Fig. 2-20 dove la sequenzializzazione procede nel modo seguente: i contenuti del program counter sono inviati all'address-bus e la cella di memoria è selezionata. Un ordine di lettura viene inviato alla memoria. Dopo un certo numero di centinaia di nanosecondi corrispondente al tempo di lettura della memoria (o *tempo di accesso*) il dato di 8 bit selezionato da questo indirizzo diventa disponibile sui pin di uscita della memoria e viene da questi propagato sul data-

bus del sistema da destra verso sinistra nella nostra illustrazione. Questo dato viene caricato internamente nell'IR della control-unit. Questo è il ciclo di fetch. Esso è ora completato.

L'IR viene poi decodificato internamente dalla control-unit, usualmente mediante una PLA (programmable-logic-array o matrice di logica programmabile). I segnali appropriati sono poi generati dalla control-unit e ciò risulta nell'esecuzione dell'istruzione.

Un'istruzione non è necessariamente lunga 8 bit. Essa può utilizzare una, due o tre parole cioè 8, 16 o 24 bit. Per una istruzione più lunga sarà necessario che la control-unit ritorni indietro alla memoria e legga il secondo e poi il terzo byte da quest'ultima. La prima parola dell'istruzione contiene sempre il suo *opcode* (o codice operativo). Decodificando il suo opcode, la control-unit apprende se deve tornare indietro alla memoria per eseguire un fetch di uno o più byte addizionali. Verranno presentati brevemente degli esempi specifici di esecuzione dell'istruzione nel caso di un microprocessore reale.

Quanto detto descrive l'esecuzione di una singola istruzione. Il problema successivo e la *sequenzializzazione automatica,* cioè l'esecuzione di istruzioni sequenziali. Questo è risolto molto semplicemente. Il program counter viene fornito di un incrementatore (o un decrementatore, in funzione della convenzione usata). Ogni volta che il PC viene utilizzato i suoi contenuti vengono automaticamente incrementati e ricaricati entro il PC. La volta successiva in cui il PC verrà utilizzato esso punterà di fatto verso la successiva parola di memoria, cioè verso l'istruzione successiva. Questo è il meccanismo automatico interno per la sequenzializzazione. L'alternativa principale all'utilizzo di questo meccanismo di sequenzializzazione automatica è chiamata *istruzione di branch.* Un programmatore sarà in grado di specificare un *branch,* o un *jump,* ad un indirizzo di memoria specificato, forzando un nuovo valore entro il program counter. Ciò verrà studiato in dettaglio nel contesto dell'8080 nel prossimo paragrafo.

La Control - Unit

Il ruolo della *control-unit* non è stato ancora descritto in dettaglio. Da un punto di vista funzionale, la control-unit ha il compito di attribuire le sequenze alle operazioni dell'intero sistema. Questo è illustrato in Fig. 2-21. La control unit genera i segnali di sincronizzazione tra la ALU, l'I/O e la memoria. Essa decodifica, preleva (esegue dei fetch) ed esegue istruzioni. Essa comunica con il mondo esterno mediante linee di input e di output: il control bus. Il control bus è gestito dalla control unit. La maggior parte delle control unit per microprocessori è realizzata usando una tecnica di *microprogrammazione.* La sequenzializzazione della control unit è ottenuta mediante un programma specializzato interno chiamato microprogramma. Il microprogramma è normalmente completamente invisibile per l'utilizzatore. Esso è realizzato mediante una ROM interna o una PLA entro il medesimo chip.

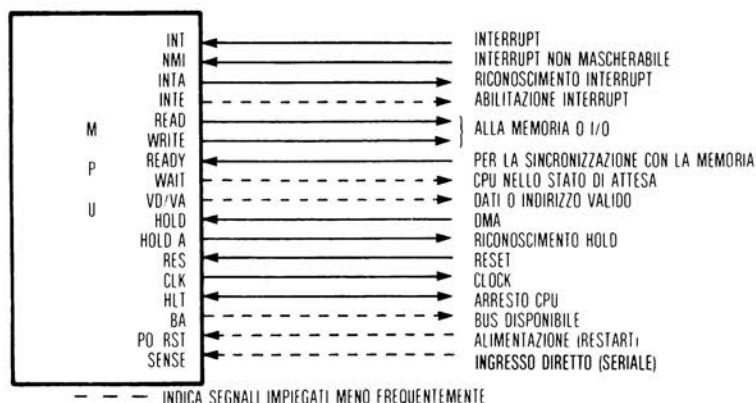



Fig. 2-22: Segnali di controllo tipici.

Gli unici tipi di chip LSI che sono normalmente microprogrammabili dall'utilizzatore sono i bit-slice. Ciò perché il controllo è esterno alle slices medesime. Per tutti gli scopi pratici i controllori microprogrammati sono essenzialmente a logica «cablata» (hardwired): essi non possono essere modificati dall'utilizzatore.

STUDIO DI UN'APPLICAZIONE CONCRETA: L'8080

Questo è un esempio didattico. Il lettore che vorrà familiarizzare con il funzionamento dettagliato di questo microprocessore, come descritto in questo paragrafo

Fig. 2-23: Architettura semplificata per l'8080.

dovrebbe essere in condizione di capire facilmente il funzionamento di qualsiasi altro microprocessore standard. I principi sono comuni a tutti i microprocessori.

L'architettura semplificata dell'8080 appare in fig. 2-23. Essa è un'architettura a bus singolo, basata su un accumulatore. Il microprocessore è dotato di sei registri general purpose che compaiono sulla destra della illustrazione. Essi sono denominati B, C, D, E, H ed L. La ragione per questa denominazione è semplice. L'accumulatore è normalmente abbreviato con «A». Gli altri registri sono perciò chiamati «B», «C», «D», «E» e così via. La ragione delle denominazioni «H» e «L» è piuttosto strana: l'8080 è il successore dell'8008. Ora l'8008 era equipaggiato di due speciali registri interni comunicanti con l'address bus, i registri «H» ed «L» (dove H sta per *High* e L sta per *Low*). L'8080 è equipaggiato con i medesimi due registri per mantenere la compatibilità. Di fatto l'8080 è fornito di due registri in più denominati «W» e «Z», che sono usati esclusivamente dalla control-unit. Questi registri appaiono in Fig. 2-24. Tuttavia essi sono completamente *invisibili* per l'utilizzatore e dovrebbero venir ignorati per il momento.

L'8080 è fornito degli usuali due registri a 16 bit che compaiono nell'angolo in basso a destra dell'illustrazione (Fig. 2-23). Essi sono il «PC» (Program Counter) e l'«SP» (Stack Pointer). Le porzioni alte e basse di questi registri sono designate H ed L ogniqualvolta sia necessario riferirsi a soli 8 bit entro questi registri. Le porzioni bassa e alta del PC saranno chiamate PCL e PCH rispettivamente. Similmente gli 8 bit inferiori e superiori dello SP saranno chiamati SPL e SPH. Questa è solo una descrizione semplificata. I registri H ed L possono essere anche manipolati come un singolo registro a 16 bit ed essi sono connessi pure all'*address-bus*.

Esamineremo una descrizione molto più dettagliata dell'8080 più avanti in questo paragrafo. Esso è illustrato in Fig. 2-24. L'illustrazione mostra che l'architettu-

Fig. 2-24: Architettura interna dell'8080.

ra dell'8080 è, di fatto, l'architettura standard che è stata descritta precedentemente.

Per permettere una descrizione più dettagliata del funzionamento di questo microprocessore bisogna presentare a questo punto alcune definizioni.

Cicli e stati dell'8080

Si è stabilito che l'esecuzione di una istruzione, per qualunque processore, inizia con una fase di *fetch*: l'istruzione deve essere portata dalla memoria in un registro speciale della control unit. Nel caso dell'8080 una operazione di fetch corrisponde ad un *ciclo-macchina* (eccettuato per l'istruzione DAD).

L'esecuzione di ciascuna istruzione richiederà da 1 a 5 cicli macchina cioè da 1 a 5 accessi in memoria. La ragione di ciò deriva dal fatto che l'istruzione può avere una lunghezza di 1,2 o 3 parole e può richiedere 1 o 2 accessi addizionali alla memoria.

Ogni ciclo macchina è realizzato internamente mediante una succesione di micro-operazioni. Ciascun passo nella sequenza è chiamato *stato interno.* Ciascun ciclo macchina richiederà da 3 a 5 di questi stati designati da S1 ad S5. Ciascun stato interno è associato, di fatto, con l'esecuzione di una *microistruzione* del microprogramma del sequenziatore. La sequenzializzazione delle micro-operazioni è sincronizzata al clock *e uno stato corrisponde alla durata di tempo fra due impulsi successivi del clock.* Più precisamente l'8080 utilizza un clock a due fasi. Le due fasi sono rispettivamente contrassegnate Φ 1 e Φ 2. La durata di una fase sarà il periodo della fase 1 del clock. L'8080A standard utilizza un clock a 2MHz.

La durata di uno stato è, perciò, di 500 ns (eccettuato per le istruzioni WAIT, HLDA e HLTA). L'esecuzione completa di un'istruzione richiederà da 4 a 18 stati. La durata di esecuzione di un'istruzione può allora essere cosi calcolata: essa varierà da 4 x 500 ns = 2000 ns = 2μs a 18 x 500 ns = 9000 ns = 9μs.

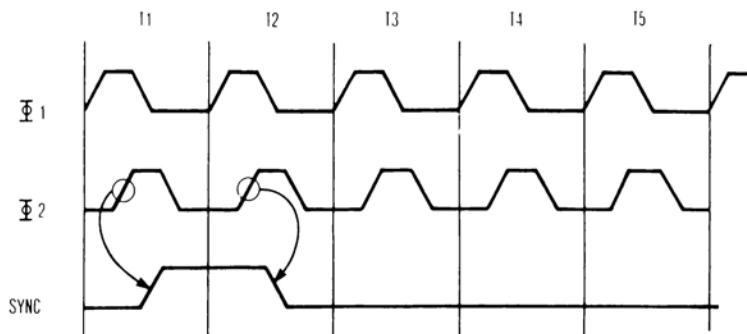



Fig. 2-25: Un clock a 2 fasi fornisce la sincronizzazione SYNC.

Esistono due versioni più veloci dell'8080A, chiamate 8080 A1 e 8080 A2. La versione più veloce usa un clock a 3MHz. Essa risulta più veloce del 33% e richiede solamente 1,3µs per un ciclo minimo (4 stati).

Formati di Istruzione

Le istruzioni dell'8080 sono elencate in Figura 2-26 e specificate in dettaglio all'Appendice C. Come per ogni altro microprocessore, le istruzioni dell'8080 possono avere un formato di 1, 2 o 3 parole. Un'istruzione specifica l'operazione da svolgere da parte del microprocessore. Da un punto di vista semplificativo, ciascuna istruzione può essere rappresentata come un Op Code seguito da un campo opzio-

NOTE:

1. Il primo ciclo di memoria (M1) è sempre un'istruzione eseguibile, durante questo ciclo viene eseguito il primo ed unico byte contenente il codice operativo.

2. Se l'ingresso READY dalla memoria non è alto durante T2 di ogni ciclo di memoria, il processore entrerà in uno stato di attesa (WAIT) (TW) fino a che READY risulta alto.

3. Gli stati T4 e T5 sono presenti, come richiesto, per i funzionamenti completamente interni alla CPU. I contenuti del bus interno durante T4 e T5 sono disponibili sul bus dati; questo è previsto soltanto per scopi di testing. Un "X" denota che lo stato è presente ma è impiegato soltanto per funzionamenti interni come la decodifica dell'istruzione.

4. Possono essere specificate solo le coppie di registri rp = B (registro B e C) oppure rp = D (registri D e E).

5. Questi stati vengono saltati.

6. La memoria legge sottocicli: verrà letta un'istruzione oppure una parola dati.

7. La memoria scrive sottocicli.

8. Il segnale READY non è richiesto durante il secondo e terzo sottociclo (M2 ed M3). Il segnale HOLD è accettato durante M2 ed M3. Il segnale SYNC non è generato durante M2 ed M3. Durante l'esecuzione di DAD sono richiesti M2 ed M3 per una somma di coppia di registri interni; non c'è riferimento alla memoria.

9. I risultati di queste istruzioni aritmetiche, logiche o di rotazione non vengono mossi nell'accumulatore A fino allo stato T2 del successivo ciclo di istruzione. Cioè A viene caricato mentre si sta eseguendo l'istruzione successiva; questa sovrapposizione consente un'elaborazione più veloce.

10. Se il valore dei 4 byte meno significativi dell'accumulatore è maggiore di 9, oppure è uguale ad 1 il bit di carry ausiliario, viene sommato 6 all'accumulatore. Se ora il valore dei 4 bit più significativi dell'accumulatore è maggiore di 9, oppure è uguale ad 1 il bit di carry, viene sommato 6 ai 4 bit più significativi dell'accumulatore.

11. Questo rappresenta il primo sottociclo (l'esecuzione dell'istruzione) del successivo ciclo di istruzione.

12. Se la condizione è soddisfatta i contenuti della coppia di registri WZ vengono fatti uscire sulle linee di indirizzo (A_{0-15}) invece del contenuto del contatore di programma (PC).

13. Se la condizione non è soddisfatta, vengono saltati i sottocicli M4 ed M5, il processore procederà immediatamente all'esecuzione di istruzione (M1) del successivo ciclo di istruzione.

14. Se la condizione non è soddisfatta, vengono saltati i sottocicli M2 ed M3, il processore procederà immediatamente all'esecuzione di istruzione (M1) del successivo ciclo di istruzione.

15. Sottociclo di lettura dello stack.

16. Sottociclo di scrittura dello stack.

17. CONDIZIONE

	CONDIZIONE	CCC
NZ	- non zero (Z=0)	000
Z	- zero (Z=1)	001
NC	- non riporto (CY=0)	010
C	- riporto (CY=1)	011
PO	- parità dispari (P=0)	100
PE	- parità pari (P=1)	101
P	- più (S=0)	110
M	- meno (S=1)	111

18. Sottociclo di I/O: il codice di selezione ad 8 bit della porta di I/O è duplicata sulle linee di indirizzo 0-7 (A_{0-7}) ed 8-15 (A_{8-15}).

19. Sottociclo di uscita.

20. Il processore permarrà nello stato di halt finche non si accetta un interrupt, un reset oppure un hold. Quando si accetta una richiesta di hold la CPU entra nel modo hold. Al termine del modo hold il processore ritorna nello stato di halt. Dopo che è stato accettato un reset il processore esegue l'istruzione forzata sul bus dati (normalmente un'istruzione restart).

SSS o DDD	Valore	rp	Valore
A	111	[illegible]	00
B	000	D	01
C	001	H	10
D	010	SP	11
E	011		
H	100		
L	101		

Fig. 2-26: Formati delle istruzioni.

MNEMONIC	OP CODE		M1[1]					M2		
	$D_7D_6D_5D_4$	$D_3D_2D_1D_0$	T1	T2[2]	T3	T4	T5	T1	T2[2]	T3
MOV r1, r2	0 1 D D	D S S S	PC OUT STATUS	PC = PC + 1	INST→TMP/IR	(SSS)→TMP	(TMP)→DDD			
MOV r, M	0 1 D D	D 1 1 0				X[3]		HL OUT STATUS[6]	DATA	→DDD
MOV M, r	0 1 1 1	0 S S S				(SSS)→TMP		HL OUT STATUS[7]	(TMP)	→DATA BUS
SPHL	1 1 1 1	1 0 0 1				(HL)	→SP			
MVI r, data	0 0 D D	D 1 1 0				X		PC OUT STATUS[6]	B2	→DDDD
MVI M, data	0 0 1 1	0 1 1 0				X			B2	→TMP
LXI rp, data	0 0 R P	0 0 0 1				X			PC = PC + 1 B2	→r1
LDA addr	0 0 1 1	1 0 1 0				X			PC = PC + 1 B2	→Z
STA addr	0 0 1 1	0 0 1 0				X			PC = PC + 1 B2	→Z
LHLD addr	0 0 1 0	1 0 1 0				X			PC = PC + 1 B2	→Z
SHLD addr	0 0 1 0	0 0 1 0				X		PC OUT STATUS[6]	PC = PC + 1 B2	→Z
LDAX rp[4]	u 0 R P	1 0 1 0				X		rp OUT STATUS[6]	DATA	→A
STAX rp[4]	0 0 R P	0 0 1 0				X		rp OUT STATUS[7]	(A)	→DATA BUS
XCHG	1 1 1 0	1 0 1 1				(HL)↔(DE)				
ADD r	1 0 0 0	0 S S S				(SSS)→TMP (A)→ACT		[9]	(ACT)+(TMP)→A	
ADD M	1 0 0 0	0 1 1 0				(A)→ACT		HL OUT STATUS[6]	DATA	→TMP
ADI data	1 1 0 0	0 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2	→TMP
ADC r	1 0 0 0	1 S S S				(SSS)→TMP (A)→ACT		[9]	(ACT)+(TMP)+CY→A	
ADC M	1 0 0 0	1 1 1 0				(A)→ACT		HL OUT STATUS[6]	DATA	→TMP
ACI data	1 1 0 0	1 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2	→TMP
SUB r	1 0 0 1	0 S S S				(SSS)→TMP (A)→ACT		[9]	(ACT)-(TMP)→A	
SUB M	1 0 0 1	0 1 1 0				(A)→ACT		HL OUT STATUS[6]	DATA	→TMP
SUI data	1 1 0 1	0 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2	→TMP
SBB r	1 0 0 1	1 S S S				(SSS)→TMP (A)→ACT		[9]	(ACT)-(TMP)-CY→A	
SBB M	1 0 0 1	1 1 1 0				(A)→ACT		HL OUT STATUS[6]	DATA	→TMP
SBI data	1 1 0 1	1 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2	→TMP
INR r	0 0 D D	D 1 0 0				(DDD)→TMP (TMP) + 1→ALU	ALU→DDD			
INR M	0 0 1 1	0 1 0 0				X		HL OUT STATUS[6]	DATA (TMP)+1	→TMP →ALU
DCR r	0 0 D D	D 1 0 1				(DDD)→TMP (TMP)+1→ALU	ALU→DDD			
DCR M	0 0 1 1	0 1 0 1				X		HL OUT STATUS[6]	DATA (TMP)-1	→TMP →ALU
INX rp	0 0 R P	0 0 1 1				(RP) + 1	→RP			
DCX rp	0 0 R P	1 0 1 1				(RP) - 1	→RP			
DAD rp[8]	0 0 R P	1 0 0 1				X		(ri)→ACT	(L)→TMP, (ACT)+(TMP)→ALU	ALU→L, CY
DAA	0 0 1 0	0 1 1 1				DAA→A, FLAGS[10]				
ANA r	1 0 1 0	0 S S S				(SSS)→TMP (A)→ACT		[9]	(ACT)+(TMP)→A	
ANA M	1 0 1 0	0 1 1 0	PC OUT STATUS	PC = PC + 1	INST→TMP/IR	(A)→ACT		HL OUT STATUS[6]	DATA	→TMP

Fig. 2-26: Formati delle istruzioni (segue).

M3			M4			M5				
T1	T2[2]	T3	T1	T2[2]	T3	T1	T2[2]	T3	T4	T5
HL OUT STATUS[7]	(TMP) →	DATA BUS								
PC OUT STATUS[6]	PC = PC + 1 B3 →	rh								
↑	PC = PC + 1 B3 →	W	WZ OUT STATUS[6]	DATA →	A					
	PC = PC + 1 B3 →	W	WZ OUT STATUS[7]	(A) →	DATA BUS					
↓	PC = PC + 1 B3 →	W	WZ OUT STATUS[6]	DATA → WZ = WZ + 1	L	WZ OUT STATUS[6]	DATA →	H		
PC OUT STATUS[6]	PC = PC + 1 B3 →	W	WZ OUT STATUS[7]	(L) → WZ = WZ + 1	DATA BUS	WZ OUT STATUS[7]	(H) →	DATA BUS		
[9]	(ACT)+(TMP)→A									
[9]	(ACT)+(TMP)→A									
[9]	(ACT)+(TMP)+CY→A									
[9]	(ACT)+(TMP)+CY→A									
[9]	(ACT)-(TMP)→A									
[9]	(ACT)-(TMP)→A									
[9]	(ACT)-(TMP)-CY→A									
[9]	(ACT)-(TMP)-CY→A									
HL OUT STATUS[7]	ALU →	DATA BUS								
HL OUT STATUS[7]	ALU →	DATA BUS								
(rh)→ACT	(H)→TMP (ACT)+(TMP)+CY→ALU	ALU→H, CY								
[9]	(ACT)+(TMP)→A									

Fig. 2-26: Formati delle istruzioni (segue).

MNEMONIC	OP CODE		M1[1]					M2		
	$D_7 D_6 D_5 D_4$	$D_3 D_2 D_1 D_0$	T1	T2[2]	T3	T4	T5	T1	T2[2]	T3
ANI data	1 1 1 0	0 1 1 0	PC OUT STATUS	PC = PC + 1	INST→TMP/IR	(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2 →	TMP
XRA r	1 0 1 0	1 S S S				(A)→ACT (SSS)→TMP		[9]	(ACT)+(TPM)→A	
XRA M	1 0 1 0	1 1 1 0				(A)→ACT		HL OUT STATUS[6]	DATA →	TMP
XRI data	1 1 1 0	1 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2 →	TMP
ORA r	1 0 1 1	0 S S S				(A)→ACT (SSS)→TMP		[9]	(ACT)+(TMP)→A	
ORA M	1 0 1 1	0 1 1 0				(A)→ACT		HL OUT STATUS[6]	DATA →	TMP
ORI data	1 1 1 1	0 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2 →	TMP
CMP r	1 0 1 1	1 S S S				(A)→ACT (SSS)→TMP		[9]	(ACT)-(TMP), FLAGS	
CMP M	1 0 1 1	1 1 1 0				(A)→ACT		HL OUT STATUS[6]	DATA →	TMP
CPI data	1 1 1 1	1 1 1 0				(A)→ACT		PC OUT STATUS[6]	PC = PC + 1 B2 →	TMP
RLC	0 0 0 0	0 1 1 1				(A)→ALU ROTATE		[9]	ALU→A, CY	
RRC	0 0 0 0	1 1 1 1				(A)→ALU ROTATE		[9]	ALU→A, CY	
RAL	0 0 0 1	0 1 1 1				(A), CY→ALU ROTATE		[9]	ALU→A, CY	
RAR	0 0 0 1	1 1 1 1				(A), CY→ALU ROTATE		[9]	ALU→A, CY	
CMA	0 0 1 0	1 1 1 1				$(\bar{A})$→A				
CMC	0 0 1 1	1 1 1 1				$\overline{CY}$→CY				
STC	0 0 1 1	0 1 1 1				1→CY				
JMP addr	1 1 0 0	0 0 1 1				X		PC OUT STATUS[6]	PC = PC + 1 B2 →	Z
J cond addr[17]	1 1 C C	C 0 1 0				JUDGE CONDITION		PC OUT STATUS[6]	PC = PC + 1 B2 →	Z
CALL addr	1 1 0 0	1 1 0 1				SP = SP - 1		PC OUT STATUS[6]	PC = PC + 1 B2 →	Z
C cond addr[17]	1 1 C C	C 1 0 0				JUDGE CONDITION IF TRUE, SP = SP - 1		PC OUT STATUS[6]	PC = PC + 1 B2 →	Z
RET	1 1 0 0	1 0 0 1				X		SP OUT STATUS[15]	SP = SP + 1 DATA →	Z
R cond addr[17]	1 1 C C	C 0 0 0			INST→TMP/IR	JUDGE CONDITION[14]		SP OUT STATUS[15]	SP = SP + 1 DATA →	Z
RST n	1 1 N N	N 1 1 1			φ→W INST→TMP/IR	SP = SP - 1		SP OUT STATUS[16]	SP = SP - 1 (PCH) →	DATA BUS
PCHL	1 1 1 0	1 0 0 1			INST→TMP/IR	(HL) → PC				
PUSH rp	1 1 R P	0 1 0 1				SP = SP - 1		SP OUT STATUS[16]	SP = SP - 1 (rh) →	DATA BUS
PUSH PSW	1 1 1 1	0 1 0 1				SP = SP - 1		SP OUT STATUS[16]	SP = SP - 1 (A) →	DATA BUS
POP rp	1 1 R P	0 0 0 1				X		SP OUT STATUS[15]	SP = SP + 1 DATA →	r1
POP PSW	1 1 1 1	0 0 0 1				X		SP OUT STATUS[15]	SP = SP + 1 DATA →	FLAGS
XTHL	1 1 1 0	0 0 1 1				X		SP OUT STATUS[15]	SP = SP + 1 DATA →	Z
IN port	1 1 0 1	1 0 1 1				X		PC OUT STATUS[6]	PC = PC + 1 B2 →	Z, W
OUT port	1 1 0 1	0 0 1 1				X		PC OUT STATUS[6]	PC = PC + 1 B2 →	Z, W
EI	1 1 1 1	1 0 1 1				SET INTE F/F				
DI	1 1 1 1	0 0 1 1				RESET INTE F/F				
HLT	0 1 1 1	0 1 1 0				X		PC OUT STATUS	HALT MODE[20]	
NOP	0 0 0 0	0 0 0 0	PC OUT STATUS	PC = PC + 1	INST→TMP/IR	X				

Fig. 2-26: Formati delle istruzioni (segue).

M3			M4			M5						
T1	T2[2]	T3	T1	T2[2]	• T3	T1	T2[2]	T3	T4	T5		
[9]	(ACT)+(TMP)→A											
[9]	(ACT)+(TMP)→A											
[9]	(ACT)+(TMP)→A											
[9]	(ACT)+(TMP)→A											
[9]	(ACT)+(TMP)→A											
[9]	(ACT)-(TMP); FLAGS											
[9]	(ACT)-(TMP); FLAGS											
PC OUT STATUS[6]	PC = PC + 1 B3 —	→ W									WZ OUT STATUS[11]	(WZ) + 1 → PC
PC OUT STATUS[6]	PC = PC + 1 B3 —	→ W									WZ OUT STATUS[11,12]	(WZ) + 1 → PC
PC OUT STATUS[6]	PC = PC + 1 B3 —	→ W	[illegible] OUT STATUS[16]	(PCH) — SP = [illegible]P - 1	→ DATA BUS	SP OUT STATUS[16]	(PCL) —	→ DATA BUS			WZ OUT STATUS[11]	(WZ) + 1 → PC
PC OUT STATUS[6]	PC = PC + 1 B3 —	→ W[13]	SP OUT STATUS[16]	(PCH) — SP = SP - 1	→ DATA BUS	SP OUT STATUS[16]	(PCL) —	→ DATA BUS			WZ OUT STATUS[11,12]	(WZ) + 1 → PC
SP OUT STATUS[15]	SP = SP + 1 DATA —	→ W									WZ OUT STATUS[11]	(WZ) + 1 → PC
SP OUT STATUS[15]	SP = SP + 1 DATA —	→ W									WZ OUT STATUS[11,12]	(WZ) + 1 → PC
SP OUT STATUS[16]	(TMP = 00NNN000) — (PCL) —	→ Z → LATA BUS									WZ OUT STATUS[11]	(WZ) + 1 → PC
SP OUT STATUS[16]	(rl) —	→ DATA BUS										
SP OUT STATUS[16]	FLAGS —	→ DATA BUS										
SP OUT STATUS[15]	SP = SP + 1 DATA —	→ rh										
SP OUT STATUS[15]	SP = SP + 1 DATA —	→ A										
SP OUT STATUS[15]	DATA —	→ W	SP OUT STATUS[16]	(H) —	→ DATA BUS	SP OUT STATUS[16]	(L) —	→ DATA BUS	(WZ) —	→ HL		
WZ OUT STATUS[18]	DATA —	→ A										
WZ OUT STATUS[18]	(A) —	→ DATA BUS										

Fig. 2-26: Formati delle istruzioni.

nale letterale o di indirizzo, comprendente una o due parole. Il campo di Op Code specifica l'operazione da svolgere. In una terminologia strettamente legata al campo dei calcolatori, l'Op Code rappresenta solamente quei bit che specificano l'operazione da svolgere, escludendo i puntatori ai registri che potrebbero essere necessari. Nel mondo dei microprocessori è conveniente definire Op Code l'operazione stessa di codificazione così come ogni puntatore su registri che essa possa incorporare. Questo «Op Code generalizzato» deve risiedere in una parola di 8 bit per ragioni di efficienza (questo è il fattore limitativo sul numero di istruzioni disponibili in un microprocessore).

Alcune istruzioni richiedono che una parola di dato segua l'Op Code. In tal caso l'istruzione sarà formata da due parole dove la seconda parola è il dato.

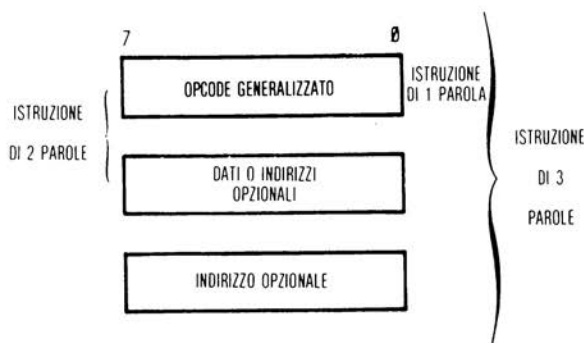



Fig. 2-27: Sequenzializzazione nell'8080.

In altri casi l'istruzione può richiedere la specificazione di un indirizzo. Un indirizzo richiede 16 bit e, perciò due parole. In quel caso l'istruzione ha 3 parole. Questo è il formato più lungo usato nell'8080 ed è il formato più lungo che sia ancora ragionevole per un microprocessore a 8 bit.

Per ciascuna parola dell'istruzione la control unit dovrà eseguire un fetch dalla memoria che richiederà due μs di tempo. Più corta è l'istruzione, più veloce risulta l'esecuzione.

Un'istruzione a parola singola

Le istruzioni a parola singola sono, in linea di principio, le più veloci e sono quelle preferite dal programmatore. Una istruzione tipica di questo per l'8080 è:

MOV r1, r2

Quest'istruzione significa: «Trasferisci i contenuti del registro r2 in r1». È una tipica operazione «da registro a registro». Ogni microprocessore deve essere fornito

con tali istruzioni che permettono al programmatore di trasferire un'informazione da uno qualsiasi dei registri della macchina in un altro. Le istruzioni che riferiscono a registri speciali della macchina come l'accumulatore o altri registri di uso particolare spesso hanno un Op Code speciale, ma non necessariamente.

Dopo l'esecuzione dell'istruzione vista sopra i contenuti di r1 saranno identici ai contenuti di r2. I contenuti di r2 *non* saranno stati modificati dalla operazione di lettura.

Ogni istruzione deve essere rappresentata internamente in un formato binario. La rappresentazione precedente è simbolica o *mnemonica*. Essa è chiamata rappresentazione di un'istruzione in *assembly-language* (linguaggio assemblatore). Essa assume semplicemente il significato di una rappresentazione simbolica conveniente della reale codificazione binaria per quell'istruzione. Il codice binario che rappresenterà questa istruzione è:

0 1 D D D S S S (bit da 0 a 7).

Questa rappresentazione è ancora parzialmente simbólica. Ciascuna lettera S o D rappresenta un bit in binario. Le tre D, D D D, rappresentano i tre bit indicanti il registro di *destinazione*. Tre bit permettono la selezione di un registro entro un massimo di 8 possibili (1 out of 8). I codici per questi registri compaiono in Fig. 2-28. Ad esempio il codice per il registro B è 0 0 0, il codice per il registro C è 0 0 1 e così via.

ABBREVIAZIONI:

B1	BYTE 1
B2	BYTE 2
B3	BYTE 3
R	REGISTRO
SSS	SORGENTE
DDD	DESTINAZIONE
RP	COPPIA DI REGISTRI
H	ALTO
L	BASSO
N	INDIRIZZO (DA 0 A 7)

CODICI:

1 - REGISTRI:

0 0 0	B
0 0 1	C
0 1 0	D
0 1 1	E
1 0 0	H
1 0 1	L
1 1 0	- (MEMORIA)
1 1 1	A

2 - COPPIE DI REGISTRI

0 0	BC
0 1	DE
1 0	HL
1 1	SP

Fig. 2-28: Codice dei registri.

In modo simile S S S rappresenta i tre bit che puntano al registro *sorgente*. La convenzione qui è che il registro 2 sia la sorgente e che il registro 1 sia la destinazione. La disposizione dei bit nella rappresentazione binaria di un'istruzione non è stata escogitata per facilitare il programmatore, ma per semplificare la sezione di controllo del microprocessore, che deve decodificare ed eseguire l'istruzione. La rappresentazione in *assembly-language*, tuttavia, è stata pensata per venire incontro al *programmatore*. Si potrebbe obiettare che MOV r1, r2 dovrebbe significare in realtà «Trasferisci i contenuti di *r1* in *r2*». Tuttavia in questo caso è stata scelta una convenzione con il fine di mantenere la compatibilità con la rappresentazione binaria.

Esercizio:

Scrivete qui sotto il codice binario che trasferirà i contenuti del registro C nel registro B. Consultate la Fig. 2-28 per i codici corrispondenti a C e a B.

Risposta:

Il codice è:............

Un altro semplice esempio di una istruzione a parola singola è: ADD r.

Questa istruzione porterà come risultato alla somma dei contenuti di un registro specificato (r) all'accumulatore (A). Simbolicamente questa operazione può essere rappresentata da: A = A+r. Si può vedere dalla Tabella 2-26 che la rappresentazione binaria di questa istruzione è:

1 0 0 0 0 S S S

dove S S S specifica il registro da sommare all'accumulatore. Ancora i codici del registro appaiono in Fig. 2-28.

Esercizio:

Qual'è il codice binario dell'istruzione che sommerà i contenuti del registro D all'accumulatore?

Risposta:

Il codice è:............

Un'istruzione a due parole

ADI dato

Questa semplice istruzione a due parole sommerà i contenuti della seconda pa-

rola dell'istruzione all'accumulatore. La seconda parola dell'istruzione (byte 2) è rappresentata simbolicamente in Tabella 2-26 da «B2». I contenuti della seconda parola dell'istruzione vengono definiti come un «literal». Essi sono dati e vengono trattati come 8 bit senza alcun significato particolare. Essi potrebbero appartenere ad un carattere o a dei dati numerici. Ciò è irrilevante per l'operazione. Dalla Fig. 2-26 si può vedere che il codice per questa istruzione è:

1 1 0 0 0 1 1 0

L'«I» nello mnemonico dell'istruzione rappresenta una operazione *immediata*. «Immediata» nella maggior parte dei linguaggi di programmazione significa che la parola successiva, o le parole, entro l'istruzione contengono un pezzo di dato che non deve essere *interpretato* (nel modo in cui si interpreta un Op Code). «I» significa che la successiva o le successive due parole devono venir trattate come un *literal* (n.d.t. letteralmente, senza interpretazione).

La control unit è programmata per «sapere» quante parole ha ciascuna istruzione. Perciò eseguirà sempre le corrette fetch ed execute sul giusto numero di parole per ciascuna istruzione. Tuttavia più è lungo il numero possibile di parole per le istruzioni, più risulta complicata la decodifica delle varie istruzioni da parte della control unit.

Un'istruzione a 3 parole

LDA addr

L'istruzione LDA addr richiede tre parole. Essa significa: «Carica l'accumulatore dall'indirizzo di memoria specificato nelle due parole successive dell'istruzione». Poichè gli indirizzi sono lunghi 16 bit, essi richiedono due parole. In binario, questa istruzione è rappresentata da:

0 0 1 1 1 0 1 0:	8 bit per l'Op Code
Low address:	8 bit per la parte inferiore dell'indirizzo
High address:	8 bit per la parte superiore dell'indirizzo

Il secondo e il terzo byte della istruzione a tre parole saranno, per convenzione, denominate con B2 e B3. L'esecuzione dettagliata delle istruzioni viste fino ad ora verrà spiegata nel prossimo paragrafo. Sarà necessario studiare solamente l'esecuzione delle istruzioni tipiche. Si sarà osservato che la Tabella 2-26 contiene di fatto la specificazione completa per l'esecuzione di qualsiasi istruzione entro l'8080. L'esecuzione delle istruzioni entro qualsiasi altro microprocessore è essenzialmente simile.

Esecuzione di istruzioni entro l'8080

In Fig. 2-24 compare una illustrazione più dettagliata della architettura interna dell'8080. Il data bus interno si trova in alto nell'illustrazione. Esso si estende verso il

mondo esterno mediante un *buffer* che lo isola dall'esterno. Le connessioni esterne a questo bus interno sono realizzate mediante alcuni pin, da D0 a D7, del package. Gli altri due bus usuali necessari per il funzionamento di un microprocessore compaiono in fondo alla figura. L'address bus ha origine dai registri di lunghezza doppia: PC, SP, HL, WZ. Per semplicità queste connessioni non sono indicate in dettaglio in figura. Il control bus è connesso essenzialmente alla control unit sulla sinistra dell'illustrazione.

In cima alla ALU compaiono gli effettivi registri di buffer. L'accumulatore (A) è seguito da un buffer chiamato ACT o «temporary accumulator». L'ingresso destro della ALU è bufferato da TMP, o «temporary register».

I *flag* non saranno considerati in dettaglio qui. Il programmatore dovrebbe rendersi conto che le operazioni svolte dalla ALU condizioneranno uno o più flag entro questo registro. Tuttavia ciò non è basilare per l'esecuzione delle istruzioni e non verrà spiegato con maggior dettaglio.

Sulla sinistra della ALU compaiono tutti i registri interni che sono stati raggruppati in un singolo blocco. La ragione di ciò è che essi svolgono di fatto una funzione identica ad una RAM interna. Essi includono i sei registri general purpose: B, C, D, E, H ed L assieme ai registri doppi: SP e PC. Inoltre B, C, D, E, H ed L sono stati raggruppati a coppie. Questo perchè l'8080 è munito di istruzioni speciali che permettono l'esecuzione di queste istruzioni speciali su *coppie di registri* come BC, DE ed HL. In più compare un gruppo speciale di registri: W e Z. Questi registri W e Z sono *indivisibili* per l'*utilizzatore*. Tuttavia essi sono necessari per il funzionamento della control unit. Di fatto *essi devono esistere in qualsiasi microprocessore*, ma generalmente non compaiono sugli schemi funzionali forniti dal costruttore non essendo visibili dall'esterno. Essi sono necessari per l'esecuzione di istruzioni interne che verranno descritte.

In testa al blocco di registri appare un multiplexer (MUX). Questo è il meccanismo di indirizzamento che selezionerà uno dei registri entro il blocco. Esso utilizza un campo di 3 bit specificati entro l'Op Code. Al di sotto del blocco di registri compare, simbolicamente, un rettangolo contenente «±1». Questo è un modo simbolico per indicare la disponibilità di un *incrementatore-decrementatore* che può automaticamente incrementare o decrementare il PC e l'SP durante alcune operazioni. Sull'estrema sinistra della figura si trova la control-unit. La control-unit è munita di un *instruction register* (IR), collegato al data bus. Il suo ruolo sarà spiegato brevemente. I contenuti di questo instruction register sono decodificati da un decoder e condurranno all'esecuzione dell'istruzione specificata.

La Tabella 2-26 presenta il funzionamento interno dettagliato di tutte le istruzioni dell'8080. Sulla sinistra compare lo *mnemonico* dell'istruzione (la sua rappresentazione in assembly-language). La colonna successiva contiene il codice binario a 8 bit per il suo Op Code, come spiegato precedentemente. La colonna successiva, contrassegnata con M1, M2, M3, M4 ed M5, contiene le operazioni dettagliate che verranno svolte

durante i successivi cicli macchina, distinti simbolicamente da M1 ad M5. Si dovrebbe ricordare a questo punto che un ciclo macchina è all'incirca corrispondente ad un fetch dalla memoria. Ciascuna istruzione richiederà almeno un ciclo macchina ed in genere più di un ciclo macchina. Ciascun ciclo macchina richiederà un certo numero di stati interni per la propria esecuzione. Durante il ciclo macchina M1 l'istruzione verrà prelevata (fetched) dalla memoria e sarà portata entro il microprocessore. Esaminiamo ora questo FETCH in dettaglio.

La fase di FETCH

La fase di FETCH di un'istruzione è realizzata durante i primi tre stati del ciclo macchina M1; essi sono chiamati T1, T2 e T3. Si può vedere in Tabella 2-26 che questi tre stati sono comuni alle istruzioni di chiamata del microprocessore poiché tutte le istruzioni devono essere fetchded prima dell'esecuzione. Il meccanismo di FETCH è il seguente:

T1: PC OUT

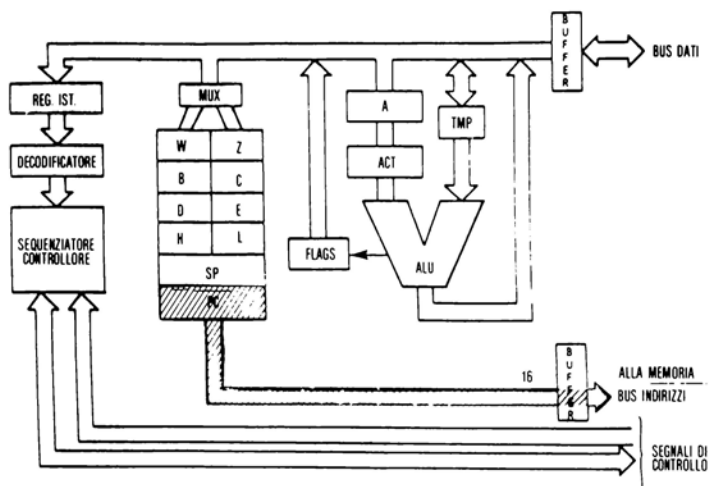



Fig. 2-29: Prelievo di istruzione: (PC) è inviato in memoria.

Il primo passo consiste nel presentare l'indirizzo dell'istruzione successiva alla memoria. Questo indirizzo è contenuto nel program counter (PC). Non appena il primo passo di ogni istruzione inizia con una richiesta di fetch, i contenuti del PC sono trasferiti sull'address bus (vedi Fig. 2-29). A questo punto viene presentato alla memoria un indirizzo e i decodificatori di indirizzo della memoria decodificheranno questo indirizzo al fine di selezionare l'appropriata locazione scelta entro la memoria. Prima che i

contenuti della locazione selezionata di memoria diventino disponibili sui pin di uscita della memoria che sono connessi al data bus, dovranno passare diverse centinaia di ns. Seguendo la prassi di progettazione standard di calcolatori, si utilizza il tempo di lettura dalla memoria per eseguire un'operazione entro il microprocessore. Questa operazione è l'incremento del program counter.

T2: PC = PC+1

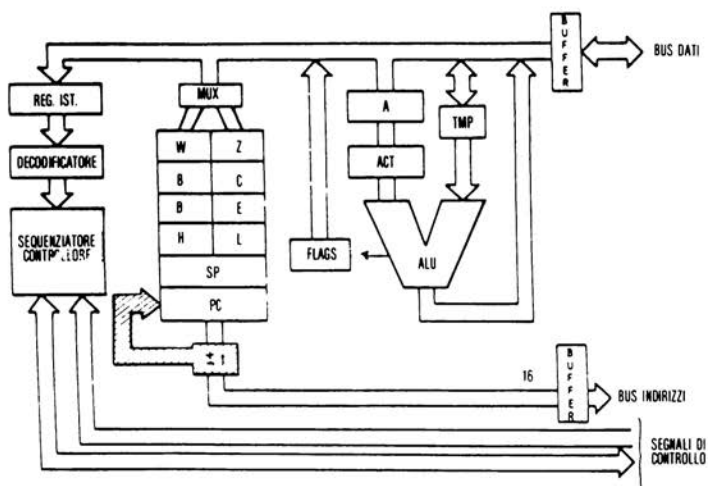



Fig. 2-30: PC viene incrementato.

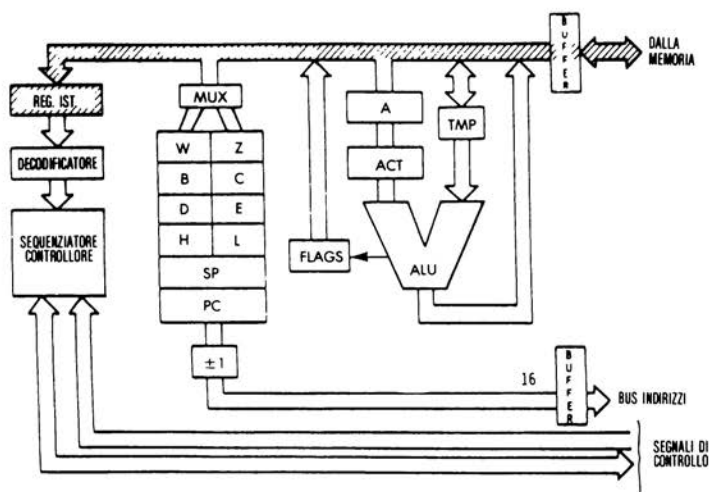



Fig. 2-31: L'istruzione arriva dalla memoria in IR.

Mentre la memoria è in fase di lettura i contenuti del PC vengono incrementati di 1 (vedi Fig. 2-30). T2 richiede 500 ns sull'8080 standard. Alla fine dello stato T2 i contenuti della memoria sono disponibili e possono ora venir trasferiti entro il microprocessore:

T3: INST in IR

Durante lo stato T3 l'istruzione che è stata letta ed estratta dalla memoria viene depositata sul data bus e di qui trasferita all'instruction register dell'8080 dove essa viene decodificata.

Si dovrebbe notare che lo stato T4, come l'M1, saranno sempre richiesti. Dopo che l'istruzione è stata depositata in IR durante T3 è necessario *decodificarla* ed *eseguirla.* Ciò richiederà almeno uno stato macchina, T4.

Un ridotto numero di istruzioni richiedono un ulteriore stato di M1 (stato T5). Questo stato non è necessariamente usato. Esso verrà skippato (n.d.t.: brutto neologismo dall'inglese skipped che significa saltato, evitato) dal processore per la maggior parte delle istruzioni. Osservando la Tabella 2-26, i riquadri vuoti nella colonna di T5 significano appunto che lo stato non è utilizzato dal processore. Ogni volta che l'esecuzione di un'istruzione richiederà più del ciclo macchina M1, ad esempio M1, M2 e più cicli, la transizione dallo stato T4 di M1 allo stato T1 di M2 avverrà direttamente.

Esempio 1: MOV D, C

Per pura coincidenza in questo esempio capita che il registro di destinazione sia denominato D. Il trasferimento è illustrato in Fig. 2-32.

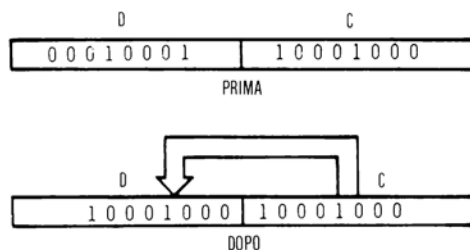



Fig. 2-32: Trasferimento di C in D.

Questa istruzione è stata descritta nel paragrafo precedente. Essa trasferisce i contenuti del registro C, denotati con «C», entro il registro D. La sequenzializzazione interna compare sulla linea 1 di Tabella 2-26.

I primi tre stati del ciclo M1 sono usati per eseguire il fetch della istruzione dalla memoria. Alla fine di T3, l'istruzione si trova in IR, l'Instruction Register, dove può esse-

re decodificata.

Durante T4: (S S S) → TMP

I contenuti di C sono depositati in TMP (vedere l'illustrazione sottostante).

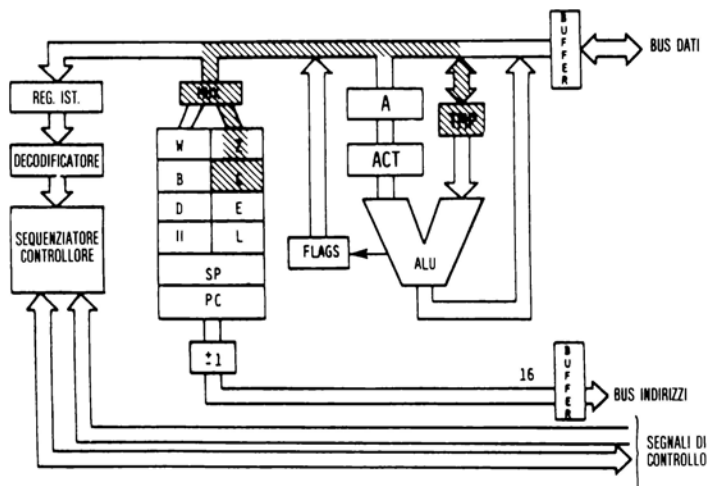



Fig. 2-33: I contenuti di C vengono depositati in TMP.

Durante T5: (TMP) → D D D

I contenuti di TMP sono depositati in D.

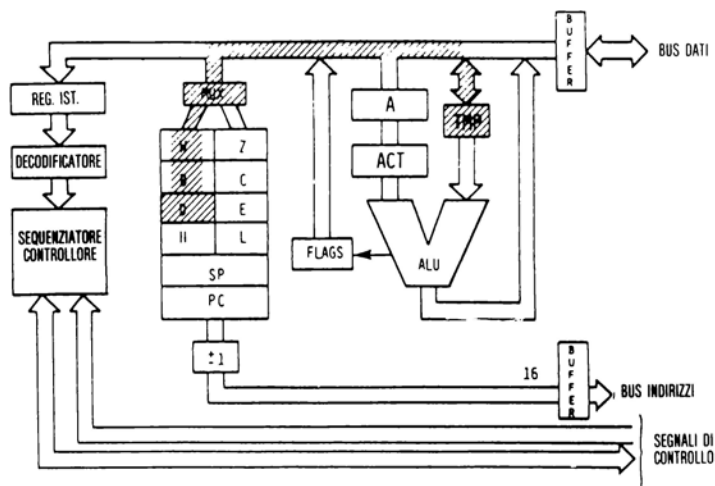



Fig. 2-34: I contenuti di TMP vengono depositati in D.

L'esecuzione dell'istruzione è ora completa. I contenuti del registro C sono stati trasferiti entro il registro di destinazione specificato, D. Ciò pone termine all'esecuzione dell'istruzione. Gli altri cicli macchina M2, M3, M4 ed M5 non saranno necessari e l'esecuzione si ferma con M1.

È possibile calcolare facilmente la durata di questa istruzione. La durata di ciascuno stato per l'8080A standard è la durata della Fase 1 del clock: 500 ns. La durata di questa istruzione è la durata di cinque stati, o meglio $5 \times 500 = 2500$ ns $= 2{,}5$ μs.

DOMANDA: *Perchè questa istruzione per trasferire i contenuti di C in D richiede due stati, T4 e T5, piuttosto che uno soltanto? Essa trasferisce i contenuti di C entro TMP e poi i contenuti di TMP in D. Non sarebbe più semplice trasferire direttamente i contenuti di C in D mediante uno stato singolo?*

RISPOSTA: *Questo non è possibile a causa del tipo di configurazione scelto per i registri interni. Tutti i registri interni sono, di fatto, appartenenti ad una singola RAM, una memoria a lettura/scrittura interna al chip di microprocessore. Ogni volta può essere selezionata o indirizzata solo una parola entro la medesima RAM (single-port o accesso singolo). Per questa ragione non è possibile leggere o scrivere contemporaneamente entro o da una RAM in due diverse locazioni. Sono richiesti invece due cicli di RAM. Si rende necessario dapprima leggere il dato dal registro RAM ed immagazzinarlo in un registro temporaneo, TMP; quindi scriverlo indietro nel registro finale di destinazione, che in questo caso è D. Ciò potrebbe essere percepito come una inadeguatezza di progetto. In effetti lo è. Tuttavia questa limitazione è comune virtualmente a tutti i microprocessori monolitici. Per risolvere il problema sarebbe necessaria una RAM dual-port. Questa limitazione non è intrinseca ai processori e normalmente non esiste nel caso dei dispositivi bit-slice. Essa è dovuta alla costante ricerca di densità logica sul chip e può essere superata in futuro.*

ESERCIZIO IMPORTANTE

A questo punto è estremamente raccomandabile che l'utilizzatore esamini da se stesso la sequenzializzazione di questa semplice istruzione prima di procedere alle più complesse. A tal fine si deve tornare indietro alla Fig. 2-24. Costruitevi alcuni «simboli» di piccole dimensioni come fiammiferi, graffette ecc. Riferendovi quindi alla Tabella 2-26, spostate i simboli sulla Fig. 2-24 per simulare il flusso dei dati dai registri entro i bus. Per esempio depositate un simbolo in PC. Ponete l'attenzione su T1. T1 sposterà il simbolo contenuto in PC al di fuori sull'address bus verso la memoria. Continuate l'esecuzione simulata in questo modo fino a che avete capito bene i trasferimenti lungo i bus e tra i registri. A questo punto dovreste aver acquistato una comprensione ragionevole del significato delle microistruzioni entro l'abaco contenuto in Tabella 2-26. Siete pronti per procedere.

Ora verranno studiate istruzioni progressivamente più complesse.

ADD r

Esaminate la 15ª riga della Tabella 2-26. Questa istruzione significa: «Somma i contenuti del registro r (specificato mediante il codice binario S S S) all'accumulatore (A) e deposita il risultato nell'accumulatore». Questa è una istruzione *implicita.* È chiamata implicita perchè non menziona esplicitamente un secondo registro. L'istruzione si riferisce esplicitamente solamente al registro r. Essa implica che l'altro registro coinvolto nell'operazione sia l'accumulatore. L'accumulatore quando è utilizzato in una tale istruzione implicita che il suo Op Code completo è lungo solamente 8 bit. Essa richiede un campo per la specificazione del registro r di soli 3 bit. Questo è un modo rapido per eseguire un'operazione di addizione.

Nel sistema esistono altre istruzioni implicite che sottointendono altri registri specializzati. Esempi più complessi di tali istruzioni implicite sono le operazioni di PUSH e di POP che trasferiranno l'informazione tra la testa dello stack e l'accumulatore e che aggiorneranno nel medesimo tempo lo Stack Pointer (SP), incrementandolo o decrementandolo. Esse manipolano implicitamente il registro SP.

L'esecuzione dell'istruzione ADD r sarà ora esaminata nel dettaglio. Dovrebbe essere seguita sulla Tabella 2-26. Questa istruzione richiederà due cicli macchina, M1 ed M2. Come al solito durante i primi 3 stati di M1 l'istruzione viene fetched dalla memoria e depositata nel registro IR. All'inizio di T4 essa viene decodificata e può essere eseguita. Si assumerà qui che il registro B venga addizionato all'accumulatore. Il codice per l'istruzione sarà allora 1 0 0 0 0 0 0 0 (il codice per il registro B è 0 0 0).

T4: (S S S) → TMP, (A) → ACT

Due trasferimenti verranno eseguiti simultaneamente. Per prima cosa i contenuti del registro sorgente specificato (in questo caso B) verranno trasferiti in TMP, cioè all'ingresso destro della ALU (vedere la Fig. 2-35). Contemporaneamente i contenuti dell'accumulatore vengono trasferiti all'accumulatore temporaneo (ACT). Ispezionando la Fig. 2-35 vi accerterete che questi trasferimenti possono avvenire parallelamente. Essi utilizzano cammini differenti entro il sistema. Il trasferimento da B a TMP utilizza il data bus interno. Il trasferimento da ACT utilizza un breve collegamento interno indipendente da questo data bus. Per guadagnare tempo entrambi i trasferimenti vengono fatti simultaneamente. A questo punto entrambi gli ingressi di sinistra e di destra della ALU sono condizionati correttamente. L'ingresso sinistro della ALU è ora condizionato con i contenuti dell'accumulatore e l'ingresso destro della ALU è condizionato dai contenuti del registro B. Siamo pronti per svolgere l'addizione. Osserviamo la Tabella 2-26. C'è una sorpresa. Se analizzate lo stato T5 di M1 ove ci aspetteremmo normalmente di vedere aver luogo l'addizione, vedrete che questo stato è semplicemente inutilizzato. L'addizione non viene eseguita!

Guardando ancora la Tabella 2-26 e spostandoci sulla destra entriamo nel ciclo macchina M2. Durante lo stato T1 non avviene niente! Esiste semplicemente un riferi-

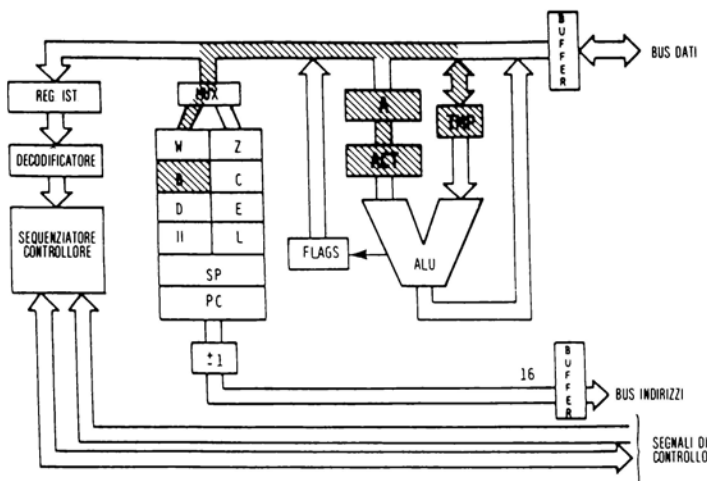



Fig. 2-35: Si verificano contemporaneamente due trasferimenti.

mento (9) nella tabella che verrà spiegato più avanti. È solamente nello stato T2 di M2 che l'addizione ha luogo:

T2 di M2: (ACT) + (TMP) → A

I contenuti di ACT sono addizionati ai contenuti di TMP e il risultato è finalmente depositato nell'accumulatore. Vedere Fig. 2-35. L'operazione è ora completata.

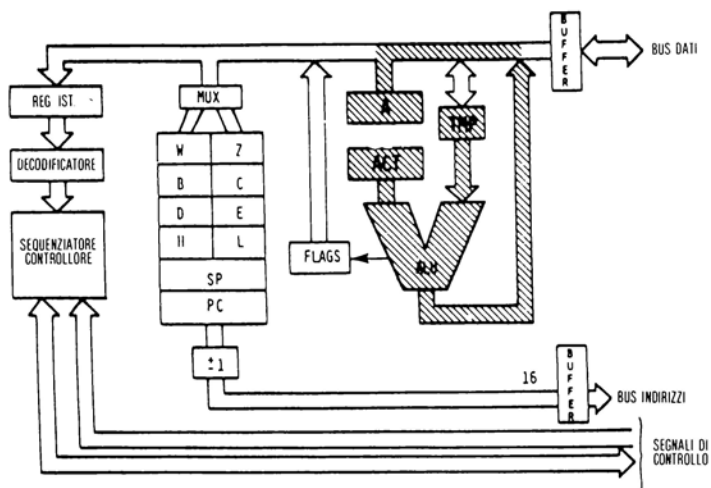



Fig. 2-36: Fine di ADD r.

DOMANDA: *Perchè il completamento dell'addizione è stato ritardato fino allo stato T2 del ciclo macchina M2 invece di venir completato durante lo stato T5 di M1?*

Questa è una domanda difficile che richiede una comprensione del progetto della CPU. Tuttavia la tecnica coinvolta è fondamentale per il progetto di CPU sincrona al clock. Tentiamo di vedere che cosa succede.

RISPOSTA: *Questo è un «trick» (espediente) standard di progettazione usato nel progetto della maggior parte delle CPU. Esso è chiamato «Fetch/Execute overlap» (o sovrapposizione del fetch con l'esecuzione). L'idea di base è la seguente: guardando indietro alla Fig. 2-36, si può vedere che la reale esecuzione dell'addizione richiede solamente l'uso della ALU e del Data Bus. In particolare essa non accederà ai registri RAM (il register block). Noi (o la control unit) sappiamo che i successivi tre stati che verranno eseguiti dopo il completamento di ogni istruzione saranno T1, T2, T3 del ciclo macchina M1 dell'istruzione successiva. Guardando indietro all'esecuzione di questi 3 stati, si può vedere che la loro esecuzione richiederà solamente l'accesso al Program Counter (PC) e l'uso dell'Address Bus. L'accesso al Program Counter richiederà l'accesso al registro RAM. (Ciò spiega che il medesimo trick non potrebbe essere usato nell'istruzione MOV r1, r2). È perciò possibile usare simultaneamente l'area tratteggiata di Fig. 2-36 e l'area tratteggiata di Fig. 2-29. Sfortunatamente c'è un fatto in più che non è stato ancora spiegato e che in qualche modo complica il quadro. Guardando indietro alla Tabella 2-26, si noterà che durante lo stato T1 di M1 sono avvenuti due fenomeni: il PC out e lo «status». Questo è dovuto alla realizzazione specifica dell'8080, che verrà ora spiegata.*

L'8080 fu il primo microprocessore standard (potente) ad 8 bit introdotto sul mercato. Ai tempi in cui fu introdotto l'Intel prese la decisione di fornire 3 alimentazioni, +5V, -5V, e +12V perchè queste erano le alimentazioni richieste dalle memorie dinamiche di allora. Sfortunatamente ciò rese inutilizzabili, se non per alimentare il chip, quattro pin invece di due come nel caso dei sistemi più aggiornati. Inoltre il clock esterno usato dal sistema 8080 richiede anch'esso diversi pin. Considerando una limitazione di fatto dei pin a 40 non era più possibile fornire tutti i pin necessari per il Control Bus!

Per questa ragione dei segnali di controllo compaiono sul Data Bus! Otto segnali di controllo sono prelevati dal Data Bus durante lo stato T1. Questo è il significato di «Status» nell'abaco nello stato T1 (Fig. 2-26). Naturalmente ciò comporta delle complicazioni aggiunte. Per l'8080 è richiesto un dispositivo speciale - l'8228 (system controller) che demultiplexi il Data Bus. Questo 8228 crea un Data Bus «pulito» che trasferisce i dati ed un Control Bus separato che trasferisce segnali di controllo. Riassumendo il Data Bus è utilizzato di fatto, durante lo stato T1 di M1, per condurre all'esterno informazioni di stato. Non può essere utilizzato per l'addizione che desideriamo eseguire. Per quella ragione diventa necessario attendere fino allo stato T2 prima che l'addizione possa essere effettivamente eseguita. Questo è ciò che avvenne nell'abaco (Fig. 2-26): l'addizione è completata durante lo stato T2 di M2.

Ora il meccanismo è stato spiegato. Quale ne è il vantaggio? Il vantaggio di questa soluzione dovrebbe ora essere chiaro. Assumiamo che avessimo realizzato uno schema diretto eseguendo l'addizione durante lo stato T5 del ciclo macchina M1. La durata dell'istruzione ADD sarebbe stata 5 × 500 = 2500 ns. Con la soluzione overlap che è stata realizzata, una volta eseguito lo stato T4 l'istruzione successiva viene immediatamente fetched. In altre parole, dopo che sono stati usati 4 stati per l'istruzione ADD, viene iniziata l'istruzione successiva. In un modo che risulterà invisibile a questa istruzione successiva, la Control Unit «intelligente» utilizzerà lo stato T2 per completare l'addizione. Sull'abaco T2 è indicata come facente parte di M2. Concettualmente M2 sarà il secondo ciclo macchina dell'addizione. Di fatto questa M2 verrà overlapped, cioè sarà identica al ciclo macchina M1 dell'istruzione successiva. Per il programmatore il ritardo introdotto da ADD sarà solamente di quattro stati, cioè 4 × 500 = 2000 ns invece di 2500 ns utilizzando la soluzione «diretta». Il miglioramento in velocità è 500 ns, o del 20%!

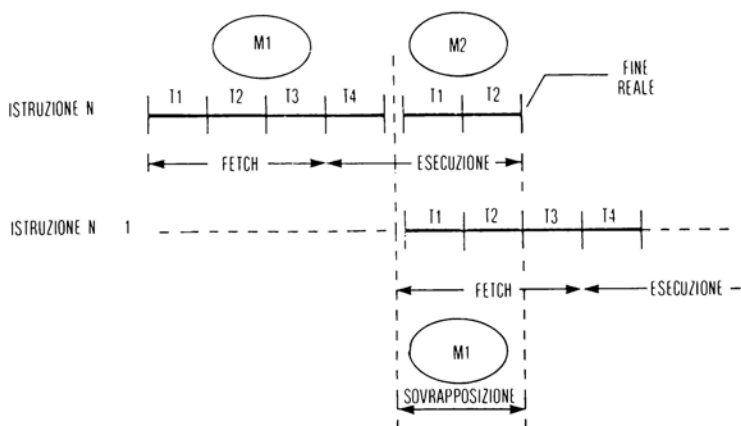



Fig. 2-37: Sovrapposizione durante T1-T2 di FETCH - EXECUTE (prelievo - esecuzione)

Questa tecnica di overlap è illustrata in Fig. 2-37. È usata ogni volta che se ne presenta la possibilità per accelerare la velocità apparente di esecuzione del microprocessore. Naturalmente non è possibile sfruttare l'overlap in tutti i casi. Devono essere disponibili a tal fine i bus o i mezzi richiesti senza che ciò crei conflitti. La control unit «sa» se è possibile un overlap e ciò è indicato con (9) sull'abaco.

DOMANDA: *Sarebbe possibile utilizzare oltre questo schema ed usare anche lo stato T3 di M2 se dovessimo eseguire un'istruzione piú lunga?*

Esamineremo ora un'istruzione più complessa.

ADD M

L'Op Code per questa istruzione è 10000110. Questa istruzione significa «somma all'accumulatore i contenuti di una locazione di memoria (M)». La locazione di memoria è specificata mediante un sistema piuttosto strano. Esso consiste nella locazione di memoria il cui indirizzo è contenuto nei registri H ed L. Questa istruzione presume che questi due registri speciali (H & L) siano stati caricati con gli opportuni contenuti prima dell'esecuzione dell'istruzione. I 16 bit di contenuto di questi registri specificherà ora l'indirizzo ove risiedono i dati nella memoria. Questi dati saranno sommati all'accumulatore e il risultato verrà lasciato nell'accumulatore.

Questa istruzione ha una storia. È stata inserita per consentire compatibilità tra il primitivo 8008 e il suo successore, l'8080. Il più vecchio 8008 non era fornito di capacità di indirizzamento diretto alla memoria! Il meccanismo previsto per accedere ai contenuti della memoria consisteva nel caricare i due registri H ed L, ed eseguire successivamente un'istruzione che facesse riferimento ad H ed L. ADD M è proprio tale istruzione. Va precisato che l'8080 non è limitato nello stesso modo dell'8008 riguardo alla capacità di indirizzamento di memoria. Esso possiede un indirizzamento diretto alla memoria. La possibilità di usare i registri H ed L diventa una prestazione aggiuntiva, non una limitazione, come avveniva nel caso dell'8008.

Segniamo ora l'esecuzione di questa istruzione sulla Tabella 2-26. Gli stati T1, T2 e T3 di M1 verranno usati, al solito, per *fetch* l'istruzione. Durante lo stato T4, i contenuti dell'accumulatore vengono trasferiti al suo registro buffer, ACT, e viene condizionato l'ingresso sinistro della ALU.

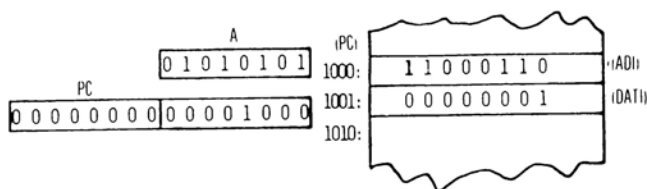
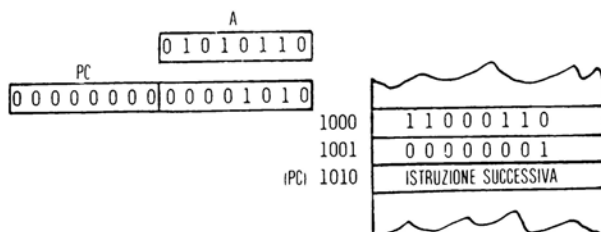



Fig. 2-38: Stato del sistema dopo l'esecuzione di ADI.

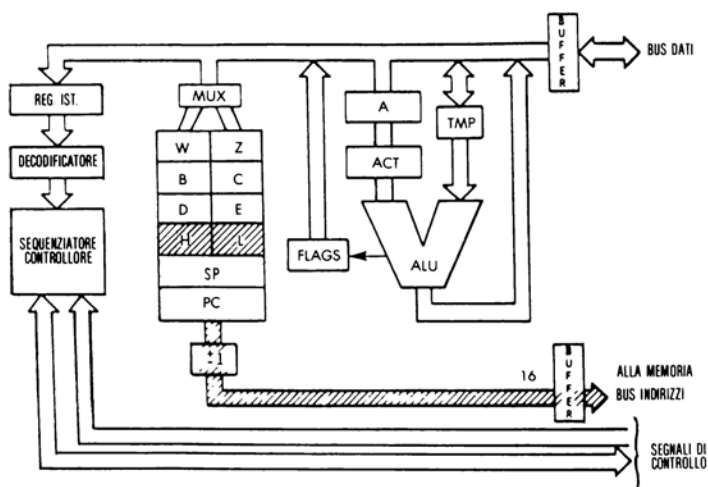



Fig. 2-39: Trasferimento dei contenuti di HL al bus indirizzi.

Per fornire il secondo byte di dati che verrà sommato all'accumulatore si deve accedere alla memoria. L'indirizzo di questo byte di dati è contenuto in H ed L. I contenuti di H ed L dovranno perciò essere trasferiti sull'address bus dal quale verranno accoppiati alla memoria. Vediamolo:

Ciclo Macchina M2: Durante il ciclo macchina M2 leggiamo: HL OUT. H ed L vengono depositati sull'address bus nello stesso modo in cui avevamo visto depositare su di esso il PC nelle precedenti istruzioni. Possiamo osservare, come è stato già indicato, che durante lo stato T1 *status* è collegato in uscita sul data bus, ma ciò non verrà utilizzato qui in alcun modo. Semplificando ci vorranno 2 stati: per la memoria per leggere i propri dati e per i dati per diventare disponibili e venir trasferiti sull'ingresso destro della ALU, TMP.

Entrambi gli ingressi della ALU sono ora condizionati. La situazione ora è analoga a quella della precedente istruzione (ADD r): entrambi gli ingressi della ALU sono condizionati. Dobbiamo semplicemente eseguire ADD come prima. Verrà usata una tecnica di overlap sulle funzioni fetch/execute e, invece di eseguire la somma nell'ambito dello stato T4 di M2, l'esecuzione finale è differita fino allo stato T2 di M3. Si può vedere in Tabella 2-24 che di fatto abbiamo durante T2: (ACT) + (TMP) → A. La somma è infine eseguita, i contenuti di ACT sono sommati a TMP e il risultato depositato nell'accumulatore A.

DOMANDA: *Qual'è il tempo apparente di esecuzione (per il programmatore) per questa istruzione? È di 7,5 µs o di 4,5 µs?*

Ora verrà esaminata una istruzione più complessa che è un'istruzione con indirizzamento diretto alla memoria facente uso dei registri invisibili W e Z:

LDA address

L'istruzione compare sulla linea 8 della Tabella 2-26. L'Op Code è 001111010. Al solito gli stati T1, T2 e T3 di M1 saranno usati per fetch l'istruzione dalla memoria. Compare ora un nuovo simbolo nello stato T4 di M1: X. Questa X indica che lo stato T4 è utilizzato, ma che non può essere descritto alcun risultato visibile. Infatti durante lo stato T4 viene decodificata l'istruzione. La control unit scopre successivamente di dover eseguire un fetch sui successivi 2 bytes di questa istruzione al fine di ottenere l'indirizzo dal quale dovrà venire caricato l'accumulatore. L'effetto di questa istruzione è di caricare l'accumulatore con i contenuti della memoria il cui indirizzo è specificato nei bytes 2 e 3 dell'istruzione. Poichè durante T4 non succede nulla di visibile si indica X. Notare che lo stato T4 è necessario per *decodificare* l'istruzione. Si potrebbe considerare ciò una perdita di tempo: solamente una parte dello stato è necessario per eseguire la decodifica. In effetti è proprio così. Tuttavia questa è la filosofia della *logica sincrona ad un orologio.* Poichè le *microistruzioni* sono usate internamente per realizzare la decodifica e l'esecuzione, questa è la penalità che deve essere pagata come corrispettivo dei vantaggi della microprogrammazione. La struttura di questa istruzione compare in Fig. 2-40.

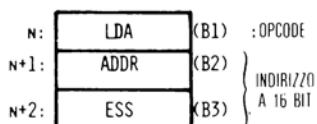



Fig. 2-40: LDA indirizzo è un'istruzione di 3 parole.

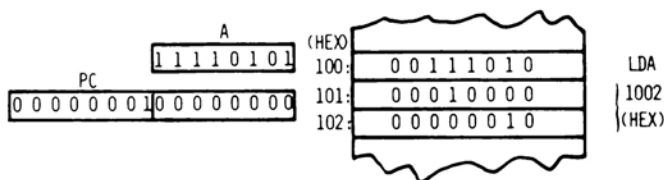



Fig. 2-41: Prima dell'esecuzione di LDA.

I successivi 2 bytes dell'istruzione verranno ora fetched. Essi specificheranno un indirizzo.

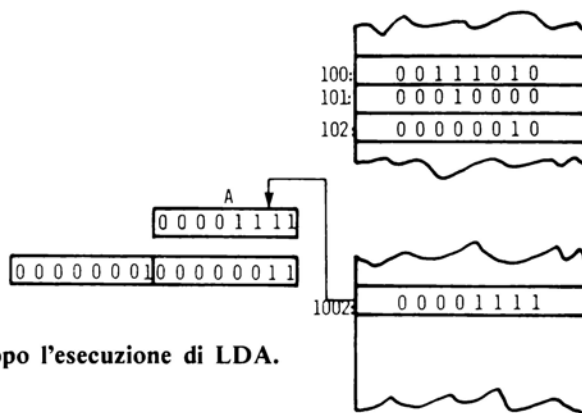



Fig. 2-42: Dopo l'esecuzione di LDA.

Secondo Ciclo Macchina M2: Al solito, i primi due stati T1 e T2 sono utilizzati per fetch i contenuti delle locazioni di memoria PC. Durante T2 il program counter PC viene incrementato. Qualche volta per la fine di T2 i dati diventano disponibili dalla memoria e compaiono sul data bus. Per la fine di T3 la parola che è stata fetched dall'indirizzo di memoria PC (B2, secondo byte dell'istruzione) è disponibile sul data bus. Essa deve ora venire immagazzinata in un registro temporaneo e depositata in Z: B2 → Z.

Ciclo Macchina M3: Di nuovo PC viene depositato sull'address bus, incrementato ed infine il terzo byte B3 viene letto dalla memoria e depositato entro il registro W del microprocessore. A questo punto, cioè alla fine dello stato T3 di M3 i registri W e Z entro il microprocessore contengono B2 e B3 cioè l'indirizzo completo a 16 bit che fu originalmente contenuto nelle due parole successive all'istruzione nella memoria. L'esecuzione può ora essere completata. W e Z contengono un indirizzo. Questo indiriz-

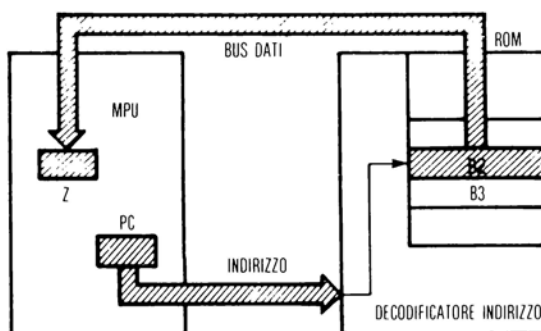



Fig. 2-43: Il secondo byte dell'istruzione va in Z.

zo dovrà essere mandato alla memoria onde estrarre i dati. Questo è fatto nel successivo ciclo di memoria:

Ciclo Macchina M4: Questa volta W e Z sono iniziati in uscita sull'address bus. L'indirizzo di 16 bit viene mandato alla memoria e entro la fine dello stato T2 i dati corrispondenti ai contenuti della locazione di memoria specificata divengono disponibili. Esso viene infine depositato in A alla fine dello stato T3. Ciò termina l'esecuzione di questa istruzione.

Quanto detto illustra l'uso di una *istruzione immediata.* Questa istruzione ha richiesto 3 byte per memorizzare un *indirizzo esplicito* di 2 byte. Questa istruzione ha richiesto inoltre 4 cicli di memoria perchè è stato necessario andare 3 volte in memoria per estrarre i 3 byte di questa istruzione a 3 parole, piú un accesso ulteriore alla memoria al fine di fetch i dati specificati dall'indirizzo. È un'istruzione lunga. Tuttavia essa è anche una istruzione di base per caricare l'accumulatore con contenuti specificati residenti in una locazione nota della memoria. Si può notare che questa istruzione richiede l'uso dei registri W e Z.

DOMANDA: *Questa istruzione avrebbe potuto usare registri diversi da W, Z nell'ambito del sistema?*

RISPOSTA: *No. Se questa istruzione avesse usato altri registri, ad esempio i registri H e L essa avrebbe modificato i loro contenuti. Dopo l'esecuzione di questa istruzione i contenuti di H, L sarebbero andati perduti. In un programma si assume sempre che un'istruzione non modificherà nessun altro registro oltre a quelli che essa utilizza esplicitamente. Un'istruzione che carichi l'accumulatore non deve distruggere i contenuti di nessun altro registro. Per questa ragione divenne necessario fornire 2 registri in più, W e Z per uso interno della control unit.*

DOMANDA: *Sarebbe stato possibile usare il PC invece di W e Z?*

RISPOSTA: *Assolutamente no. Questo sarebbe stato un suicidio. Il lettore dovrebbe analizzare ciò.*

Ora verrà studiato un altro tipo di istruzione: una istruzione di *branch* o *jump* che modifica la sequenza nella quale sono eseguite le istruzioni entro il programma. Fino ad ora abbiamo assunto che le istruzioni fossero eseguite sequenzialmente. Esistono istruzioni che permettono al programmatore di saltar fuori (jump) dalla sequenza verso un'altra istruzione entro il programma, o in termini pratici, di saltare ad un'altra area della memoria contenente il programma, o ad un altro indirizzo. Una istruzione di questo tipo è:

JMP addr

Questa istruzione appare sulla linea 18 di Tabella 2-26. La sua esecuzione verrà descritta segnando la linea orizzontale della Tabella. Questa è ancora una istruzione a 3 parole. La prima parola è Op Code e contiene 11000011. Le due parole successive contengono l'indirizzo a 16 bit verso il quale verrà eseguito il salto. Concet-

tualmente l'effetto di questa istruzione è di rinpiazzare i contenuti del Program Counter con i 16 bit che seguono l'Op Code «JUMP». In pratica per ragioni di efficienza verrà realizzato un approccio un po' diverso.

Come prima i primi 3 stati di M1 corrispondono all'istruzione di fetch. Durante lo stato T4 l'istruzione è decodificata e non si registrano altri interventi (X). I due cicli macchina successivi sono usati per fetch i byte B2 e B3 dell'istruzione. Durante M2, B3 viene fetch e depositato entro il registro interno Z. Durante M3, B3 viene fetch e depositato entro il registro interno W. Se ci riferiamo alla Tabella 2-26 avviene qualche cosa di sorprendente: sembrerebbe che l'istruzione avesse terminato l'esecuzione alla fine di T3 di M3! In effetti non è proprio così. Riferendoci alla parte destra dell'abaco possiamo vedere in un inserto 2 stati in più. Questi saranno infatti i 2 stati successivi da realizzare per il processore durante le successive istruzioni di fetch, come avviene già nel caso della addizione. I 2 passi successivi indicati verso destra sulla Tabella 2-26 verranno eseguiti al posto dei passi usuali per T1 e T2 dell'istruzione successiva. Esaminiamoli.

I due passi successivi saranno: WZ OUT: e (WZ) + 1 → PC. In altre parole i contenuti di WZ saranno usati al posto dei contenuti di PC durante la successiva istruzione di fetch. La control unit avrà registrato il fatto che è stato eseguito un jump ed eseguirà l'inizio dell'istruzione successiva in modo differente.

L'effetto di questi due stati anomali è il seguente:

L'indirizzo posto sull'address bus del sistema sarà l'indirizzo contenuto in W e Z. In altre parole l'istruzione successiva sarà fetch dall'indirizzo che era contenuto in W e Z. Questo è effettivamente un *branch*. In più i contenuti di WZ verranno incrementati di 1 e depositati nel program counter, in modo che l'istruzione successiva venga fetch correttamente utilizzando normalmente il PC. L'effetto è perciò corretto.

DOMANDA: *Perchè non sono stati caricati direttamente i contenuti del PC? A quale scopo sono stati utilizzati i registri intermedi W e Z?*

RISPOSTA: *Non è possibile utilizzare il PC. Se avessimo caricato la parte inferiore del PC (PCL) con B2, invece di utilizzare Z, avremmo distrutto il PC! Sarebbe stato allora impossibile eseguire un fetch di B3.*

DOMANDA: *Sarebbe stato possibile utilizzare solamente Z, invece di W e Z?*

RISPOSTA: *Sì, ma sarebbe stato più lento. Avremmo potuto caricare Z con B2, quindi fetch B3 e depositarlo nella metà superiore del PC (PCH). Tuttavia sarebbe poi stato necessario trasferire Z entro PCL prima di utilizzare i contenuti del PC. Ciò avrebbe rallentato il procedimento. Per questa ragione si devono usare entrambi W e Z. In più, per risparmiare tempo, W e Z non vengono trasferiti nel PC. Essi sono direttamente accoppiati all'address bus per eseguire il fetch dell'istruzione successiva. La comprensione di questo punto è cruciale per capire l'esecuzione efficiente delle istruzioni entro il microprocessore.*

DOMANDA: *(solamente per il lettore attento ed informato): Che cosa succede quando viene richiesto un interrupt alla fine di M3?* (Se l'esecuzione dell'istruttore è sospesa a questo punto il program counter punta all'istruzione successiva al jump e l'indirizzo di jump contenuto in W e Z andrà perduto).

La risposta è lasciata come interessante esercizio per il lettore attento.

Altre Istruzioni

Seguire l'esecuzione di altre istruzioni su questo abaco osservando contemporaneamente i trasferimenti di dati sullo schema dell'8080 è un esercizio molto utile.

Le istruzioni raccomandate sono:

RAL (Rotate Accumulator Left), CMP r (paragona i contenuti dell'accumulatore con i contenuti del registro r), e le istruzioni PUSH o POP, che compaiono in Tabella 2-26. La sequenzializzazione dovrebbe essere facile da seguire a questo punto.

Sommario di Studio dell'8080

Abbiamo descritto ora l'esecuzione dettagliata dei principali tipi di istruzioni disponibili in un microprocessore «standard», l'8080. Il lettore interessato dovrebbe essere capace ora di seguire l'esecuzione virtualmente di qualsiasi altra istruzione entro il microprocessore, con l'aiuto dell'abaco allegato. L'architettura interna della maggior parte dei microprocessori standard è essenzialmente simile ed altrettanto lo è l'esecuzione.

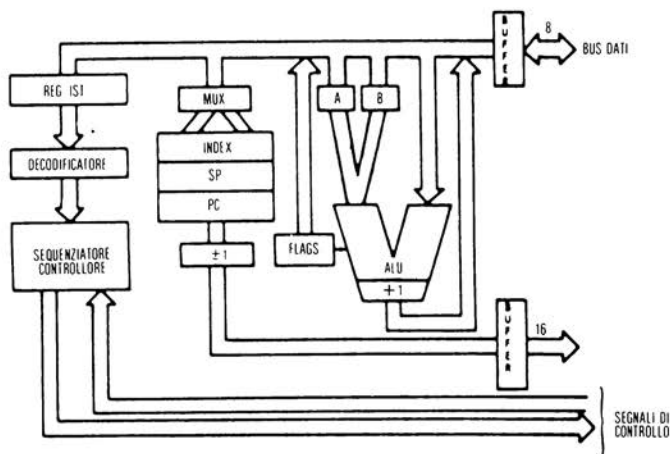



Fig. 2-44: Architettura del 6800 della Motorola.

Come esempio l'architettura del Motorola 6800 appare in Fig. 2-44. Il 6800 ha due accumulatori (A e B) e un registro indice (IX). Tuttavia esso non ha i registri di impiego generale dell'8080 (B, C, D, E, H, L). Si può dire semplicemente qui che la funzionalità di questi due microprocessori è essenzialmente simile.

ARCHITETTURE INTERNE DEL MICROPROCESSORE

Da un punto di vista funzionale esistono tre limitazioni di base sulla realizzazione di architetture di microprocessori. Esse sono:

1. Limitazione a 40 pin

Potrebbe essere sorprendente scoprire che una delle principali limitazioni poste sul progetto di componenti LSI sia la limitazione a 40 o 42 pin. Questa limitazione deriva da considerazioni economiche dovute primariamente al fatto che i tester industriali non accetterebbero packages aventi più di 40 ÷ 42 pin. Sono disponibili componenti aventi un maggior numero di pin (ad esempio 64), ma essi non sono facilmente provabili da parte di un utilizzatore industriale. Non ci si aspetta che questa limitazione venga tolta nel prossimo futuro. Esaminiamo l'impatto di questa limitazione.

- Il microprocessore deve essere dotato di un data bus. Il data bus richiederà otto pin (vedi Fig. 2-45).
- Esso richiederà un address bus a 16 bit (vedi Fig. 2-46). Si utilizzano così 16 pin in più. Totale: 24.

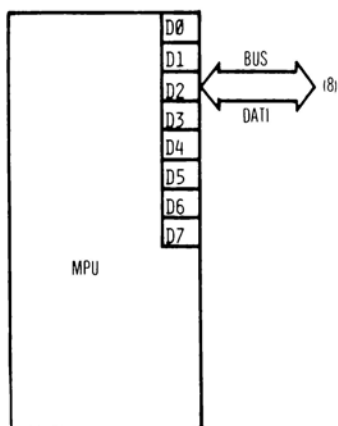



Fig. 2-45: Il bus dati richiede 8 pins.

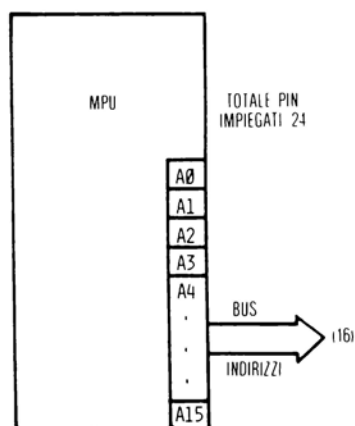



Fig. 2-46: Il bus indirizzi richiede 16 pins.

– Esso richiederà almeno due pin per l'alimentazione (vedi Fig. 2-47). Totale 26.
– Richiederà almeno due pin per il collegamento ad un cristallo esterno o ad un orologio (vedi Fig. 2-48).

Totale sino a qui: 28 pin minimo e generalmente 30.

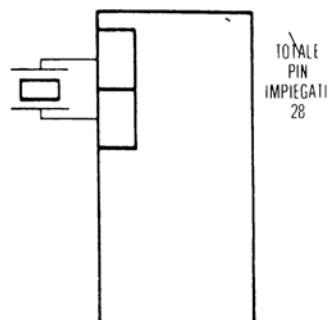
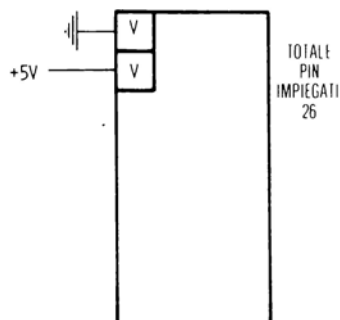



Fig. 2-47: L'alimentazione richiede 2 pins. Fig. 2-48: Il clock richiede 2 pins.

A questo punto sono disponibili solamente da 10 a 12 pin per completare il progetto. Questi pins sono necessari per il control bus (Fig. 2-49). In Fig. 2-20 compaiono i segnali tipici usati o necessari per il control bus. Si può facilmente vedere che la complessità del control bus è seriamente limitata dalla mancanza di pins disponibili. Dieci pins sono un minimo assoluto per una sincronizzazione con gli eventi esterni. In molti normali sistemi utilizzanti un calcolatore, non è raro utilizzare 40 o 50 linee per il controllo.

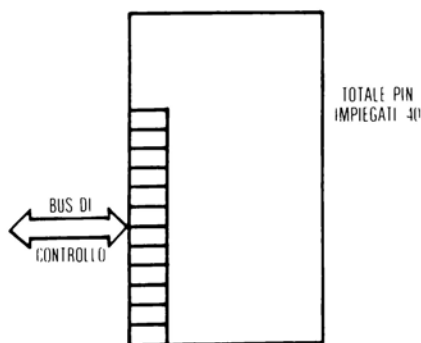



Fig. 2-49: Il bus di controllo richiede 10-12 pins.

In pratica, ciò significa *che microprocessori a 16 bit non possono essere realizzati in modo efficiente su un package a 40 pin.*

Un microprocessore a 16 bit realizzato su un package a 40 pin non può comunicare con il mondo esterno mediante un bus a 16 bit. Per prima cosa il data bus dovrà essere *multiplexato,* la parte inferiore del dato verrà trasferita in ingresso (o in uscita) e poi la parte superiore. Questo significa spesso che le istruzioni (della lunghezza di 16 bit) dovranno essere portate entro il microprocessore in due operazioni successive. Un tale microprocessore a 16 bit non sarà significativamente più veloce di uno ad 8 bit, e, in compenso sarà notevolmente più complesso da utilizzare. Per questa ragione, questo approccio non viene molto probabilmente ricercato. L'approccio «pulito» per progettare un microprocessore a 16 bit consiste nell'usare un maggior numero di pin. La Texas Instruments ha tenuto conto proprio con il TMS 9900. Tuttavia, esso richiede *64 pin,* e potrebbe non conquistare un'ampia diffusione a causa di ciò. Bisogna precisare che ci stiamo riferendo qui a microprocessori su un unico chip. Non a *microcomputers* su un unico chip. Verrà precisato più avanti che il prossimo prodotto standard che diverrà disponibile sarà probabilmente un *microcomputer* a 16 bit (non un microprocessore) su di un unico chip. La ragione è che la memoria sarà già entro il chip. Nessun address bus esterno sarà necessario e 16 pin in più diventeranno disponibili per l'I/O: il microcomputer a 16 bit su un unico chip sarà in grado di comunicare con l'esterno mediante un data bus di 16 bit o anche più ampio e richiederà ciò nonostante solamente 40 pin.

2. Area del Chip

Il secondo vincolo fondamentale dovuto ad una limitazione tecnologica è l'area massima del chip che può essere realizzata economicamente in ogni momento. Considerando i limiti sulla densità di integrazione che possono essere ottenuti, il numero di funzioni che devono venir realizzate nell'area limitata del die dovrebbe essere il più alto possibile. Questo è un vincolo essenziale nel progetto. È stato indicato che questo vincolo ha condotto alla realizzazione di sistemi a singolo bus, basati sull'uso di un accumulatore e all'utilizzo di control unit microprogrammate, così come a RAM singole interne per contenere i registri. La complessità delle operazioni realizzate dalla ALU è naturalmente ridotta dalla dimensione del die.

È ridotta inoltre dal campo di 8 bit utilizzabile correntemente per gli Op Code. Considerando il piccolo numero di bit disponibili, la maggior parte dei microprocessori non può fornire più di 60 ÷ 80 istruzioni realmente differenti tra loro. Mano a mano che le aree dei chip si espandono in futuro, diventerà realistico considerare altre architetture, in particolare architetture più efficienti che condurranno al risultato di velocità molto maggiori ed a funzioni più potenti.

3. Velocità della Tecnologia

Il terzo vincolo è dovuto alla velocità limitata delle tecnologie correntemente utilizzate per ottenere la densità richiesta. Queste tecnologie sono essenzialmente la

NMOS (la più usata), la PMOS (la più vecchia e perciò utilizzata molto frequentemente, sebbene sia più lenta) e la CMOS (utilizzata nelle applicazioni spaziali che richiedono bassi consumi di potenza).

La velocità massima è correntemente di un microsecondo per l'esecuzione di una tipica istruzione. Nella tecnologia si dovranno raggiungere progressi fondamentali per ottenere tempi di esecuzione più rapidi. Si potrebbero ottenere alcuni progressi mediante variazioni di architettura, come ad esempio dotando la ALU di un registro Q o di estensione per la ALU da usarsi per la moltiplicazione.

Le tecnologie più veloci, per la maggior parte bipolari, conducono ad una densità molto più bassa e non permettono la realizzazione del microprocessore completo su di un unico chip. Esse sono limitate ai bit-slices.

Come risultato delle limitazioni precedenti, sono emerse quattro architetture fondamentali per i microprocessori.

LE QUATTRO ARCHITETTURE PRINCIPALI

Le quattro architetture di base per i microprocessori sono illustrate in Fig. 2-50.

1. Architettura Standard

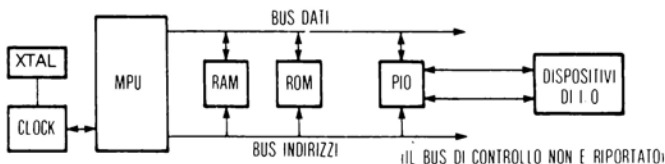
L'architettura di un microprocessore standard è caratterizzata da una MPU la quale incorpora in un chip le funzioni di una CPU tradizionale.
La memoria e l'I/O sono esterni al chip. Fino alla fine del 1976, infatti, le MPU non sono state in grado di fornire tutte le funzioni di una CPU entro un unico chip. Esse richiedevano almeno un clock esterno, più il suo cristallo di quarzo. Oggi, tutti i nuovi progetti incorporano un clock entro la medesima MPU. Il quarzo, a causa del suo ingombro, non può essere incorporato sul chip.

L'architettura standard è caratterizzata dai tre bus che abbiamo descritto. Nell'illustrazione di Fig. 2-50 sono mostrati solamente il data bus e l'address bus. In particolare nell'architettura standard tutti gli altri componenti del sistema saranno collegati ai bus. L'illustrazione in testa alla Fig. 2-50 mostra la Read Only Memory (ROM), la Random Access Memory (RAM) e un tipico chip di interfaccia, un PIO (Parallel Input-Output), collegato all'I/O (periferiche).

2. Microcomputer ad 1 Chip

Il progresso dell'integrazione LSI permette ora la realizzazione di tutti i componenti dei sistemi su un chip singolo. Fino dal 1976 è possibile avere una MPU, un clock, una ROM ed una RAM su un unico chip. In più tali chip incorporano delle possibilità aggiunte come un timer programmabile e un circuito di re-start. Il quarzo è ancora esterno al chip. La conseguenza importante di questo fatto dal punto di vista dell'architettura è che non è più necessario fornire un address bus generale. Le 16 linee precedentemente riservate per l'address bus, assieme ad alcune linee di controllo, sono ora

disponibili. Un microcomputer ad 1 chip rende disponibili per le comunicazioni con l'esterno almeno tre bus di I/O di 8 bit. Queste linee possono essere collegate direttamente a dispositivi di I/O.

DOMANDA: *Perchè sono disponibili almeno 24 linee per l'I/O su un microcomputer ad 1 chip?*

2- MICROCOMPUTER SU CHIP SINGOLO

XTAL

LINEE SENSORI

BUS 1

BUS 2

BUS 3

DISPOSITIVI DI I/O

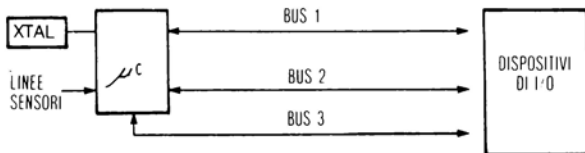
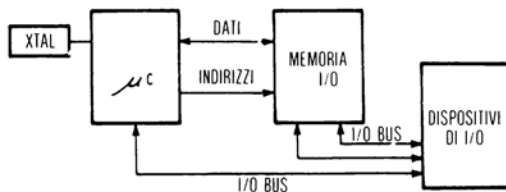
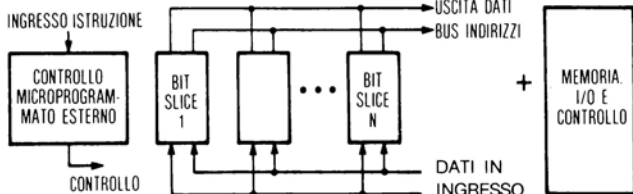



Fig. 2-50: Le 4 architetture di base.

RISPOSTA: *Il precedente data bus e il precedente address bus ora sono disponibili: 8+16 = 24 bit, che sono utilizzabili. Inoltre alcuni segnali di controllo diventano superflui, quindi si rendono disponibili un totale di 26 o 27 linee di I/O. Le caratteristiche dei microcomputers a 1 chip verranno presentate al Capitolo 4.*

3. Microcomputer a 2 Chip

Concettualmente il microcomputer a 2 chip sta a metà strada tra l'architettura standard e il microcomputer ad 1 chip. In un sistema di questo tipo il chip MPU è essenzialmente analogo al microprocessore standard, ma esso incorpora anche il clock più qualche capacità di I/O diretto. La differenza significativa sta dalla parte della memoria – I/O. Vengono forniti uno o più componenti speciali che sono forniti di ROM, RAM e di I/O su un singolo chip. Diventa allora possibile costruire un microcomputer ad 1 chip. Le dimensioni tipiche correnti sono di 2K × 8 per la ROM, di 512 × 8 per la RAM. Considerando la dimensione limitata della memoria, l'address bus necessario per specificare gli indirizzi può essere ridotto. Le linee liberate dall'address bus diventano disponibili per altre funzioni. In un tale sistema lo stesso chip di MPU è tipicamente fornito di almeno 8 linee di I/O diretto. Il chip di memoria più I/O è anch'esso fornito di almeno 8 fino a 16 di tali linee. Il sistema a 2 chip possiede allora una capacità di I/O diretto di 24 bit.

Sebbene un tale sistema possa sembrare equivalente al microcomputer ad 1 chip da un punto di vista funzionale, esso è molto diverso da un punto di vista dell'efficienza. La differenza principale sta nel fatto che un chip di microprocessore può risiedere sul chip completo, senza suddividere e attribuire parte di quest'ultimo alla memoria o all'I/O. Questi sistemi sono caratterizzati da un'alta velocità del microprocessore e da funzioni complesse. Tipicamente complessità ed efficienza sono essenzialmente identiche ad un microprocessore standard. La differenza principale sta nel fatto che la memoria e l'I/O che comparivano sul lato destro dell'illustrazione in testa a Fig. 2-50 sono ora stati riuniti su un singolo chip. D'altra parte i microcomputer ad 1 chip sono ancora limitati riguardo alla complessità dei loro set di istruzione e della loro efficienza.

I mercati per queste tre architetture sono molto differenti: il mercato principale per i microprocessori standard è il mercato che richiede programmi complessi o alta efficienza. Il mercato per i microcomputer a 1 chip è quello dei semplici controlli industriali che coinvolgono grandissimi quantitativi (oltre 100.000) considerando la necessità di mascherare la ROM sul medesimo chip. Il mercato per i microcomputer a 2 chip è il mercato delle applicazioni che richiedono un'alta efficienza, ma solamente una dimensione limitata di memoria. Esso è intermedio tra i due mercati precedenti.

4. I Bit-Slice

I bit-slice non dovrebbero esser chiamati microprocessori. Essi sono stati qualificati come microprocessori perchè sono dispositivi LSI. Tuttavia essi sono fonda-

mentalmente diversi. Sarebbe più corretto chiamarli dispositivi bit slice o processori bit slice poichè sono slice (fette) di un processore. La realizzazione del processore con dei bit slice è molto semplice. Fondamentalmente le slice vengono affiancate per costituire un'ALU con i suoi registri. Un'unità di riporto in avanti (carry - look - ahead) può venir aggiunta per aumentare l'efficienza aritmetica. Alcuni gate in più possono ancora venir aggiunti per aumentare ulteriormente la velocità delle operazioni di moltiplicazione e divisione. La vera complessità nello sviluppare un computer bit slice non risiede nella realizzazione della control unit e della circuiteria associata come i contatori ad anello, le decodifiche, le logiche condizionali, i multiplexer e la gestione del bus.

Una ALU con i suoi registri può essere costituita da quattro a sei package. La realizzazione di una completa control processing unit richiede da 20 a 50 package. Il mercato dei dispositivi bit slice è il mercato delle CPU veloci. Questa è la tecnologia LSI più veloce. Tutti i bit slice costruiti fino ad oggi sono bipolari. Una istruzione tipica sarà eseguita in un tempo da 100 a 200 ns virtualmente per ogni numero di bit.

SOMMARIO

Sono state descritte le principali architetture per microprocessori. È stato indicato che l'architettura dominante è quella a bus singolo, fondata sull'uso di un accumulatore. La struttura interna di un microprocessore standard è stata poi studiata nel dettaglio. Sono stati descritti: l'ALU, i flag, i registri, i bus, lo stack e lo scratch pad. Ci si è soffermati sul meccanismo per l'esecuzione delle istruzioni e la loro sequenzializzazione. È stato presentato uno studio completo per una applicazione dell'8080. È stata studiata passo per passo l'effettiva sequenzializzazione di ciascun tipo principale di istruzioni. A questo stadio, il lettore che abbia studiato attentamente i contenuti di questo capitolo, dovrebbe essersi completamente familiarizzato con le operazioni interne di un microprocessore, con le ragioni delle scelte principali che sono state fatte e con la loro realizzazione. I bus creati dal microprocessore verranno ora connessi ai componenti esterni al fine di costruire progressivamente un sistema che verrà infine completato al Capitolo 5. Gli altri componenti LSI necessari per metter insieme un sistema verranno ora esaminati in dettaglio nel prossimo capitolo.

CAPITOLO 3

COMPONENTI DEL SISTEMA

LE FAMIGLIE DI MICROPROCESSORI

I microprocessori oggi formano delle famiglie. Sono stati presentati diversi componenti LSI che possono essere collegati direttamente ai bus di microprocessori specifici. Sfortunatamente la maggior parte di essi non è intercambiabile. Quando furono introdotti i primi microprocessori non era virtualmente disponibile alcun altro circuito. Esistevano memorie (ROM, RAM) e UART. Il lettore ricorderà che l'8080, in particolare, richiede un de-multiplexing del proprio data bus. Un tale componente oggi esiste; è l'8228. Al tempo dell'introduzione dell'8080 esso non esisteva. La ragione del ritardo nello sviluppo di componenti di supporto ai microprocessori è molto semplice. All'inizio nessuno credette che ci sarebbe stato un mercato significativo. Oggi ognuno si rende conto che un microprocessore non può essere venduto da solo e che richiede un'ampia famiglia di componenti per essere utilizzabile. Tutti i principali costruttori di componenti oggi forniscono una famiglia completa di componenti per ciascun tipo di microprocessore. Passeremo in rassegna qui tutti i tipi importanti di componenti. Dopo questa breve presentazione, ciascuno di essi verrà studiato in dettaglio.

I TRE COMPONENTI DI BASE DEL SISTEMA

I tre componenti di base del sistema sono:

1. Il *microprocessore* stesso (MPU) e qualsiasi componente addizionale richiesto di «supporto», come il clock e il quarzo.
2. *La memoria.* Questa include sia la ROM (per il programma) e la RAM (per i dati).
3. *I chip di interfaccia I/O.* Questi possono consistere in una UART se è necessaria una conversione serie/parallelo o una PIO, se è necessaria una interfaccia parallelo/parallelo.

Usualmente sono necessari dei circuiti di supporto. Essi sono essenzialmente *latch* e *driver.*

I *latch* sono usati ogni volta che sia necessario preservare o congelare l'informazione. Tipicamente i latch sono usati per interfacciarsi a dispositivi di I/O. Il più delle volte essi sono già contenuti nel chip di interfaccia di input-output.

I *driver* sono necessari considerando la limitazione di capacità di pilotaggio del bus, da parte dei chip MOS. Un microprocessore essendo un dispositivo MOS può al

massimo pilotare un carico TTL. Se si devono collegare ai bus più di cinque o al massimo sette chip (n.d.t. della medesima famiglia), diventa necessario aggiungere dei bus driver sul data bus, l'address bus e il control bus.

I CHIP DI I/O MANAGEMENT

Timer ad Intervallo Programmabile (PIT)

La maggior parte dei programmi su microprocessori deve generare dei ritardi. I ritardi sono usati sull'uscita, oppure sull'ingresso, ad esempio per misurare la lunghezza del tempo intercorrente tra due impulsi successivi. Eseguire un conteggio a programma è semplice, ma inefficiente in termini di tempo di occupazione del processore. Ogni qualvolta tali esigenze di conteggio «sprecano» il tempo del processore che è necessario per altre funzioni, è possibile sostituire i programmi software mediante un circuito hardware. Questo è il ruolo del PIT. Il PIT si comporta come un contatore multiplo, utilizzabile sia in ingresso che in uscita. Sull'input, esso può essere usato per misurare la durata del tempo trascorso (elapsed time) tra due impulsi successivi, oppure la durata di un impulso. Sull'output, esso può essere usato come un contatore di tempo che generi un ritardo programmato. L'uso del PIT aumenterà significativamente la quantità di tempo disponibile per altre attività del microprocessore. Il suo svantaggio essenziale è il fatto di introdurre un chip addizionale nel sistema. Il suo uso è, perciò, generalmente limitato a sistemi che non verranno prodotti in grandi quantitativi, cioè non «limitati sul numero di chip».

In particolare il PIT è necessario per un controllo di processo in *real time:* gli interrupts impediscono di mantenere una temporizzazione accurata del conteggio.

L'affermazione che il microprocessore «sprechi» tempo nei conteggi può non essere valida. Ogni volta che la velocità del microprocessore è sufficiente per eseguire tutte le funzioni richieste nonostante il conteggio a software questo è un utilizzo perfettamente valido del microprocessore. Si deve ricordare che il microprocessore è un componente di sistema relativamente economico. È perfettamente accettabile dedicare il tempo del microprocessore a compiti di «basso livello». Questo nuovo concetto è contrario alla filosofia precedente di ottimizzare i tempi dei processori.

Controllore Programmabile di Interrupt (PIC)

Il PIC è un nuovo componente che permette l'uso di linee multiple di interrupt con un microprocessore standard. La maggior parte dei microprocessori sarà in grado di gestire una o due linee di interrupt (gli interrupt verranno spiegati in dettaglio più avanti in questo capitolo). Alcuni sistemi richiedono la disponibilità di otto o più linee di interrupt. Il PIC eseguirà la gestione (management) delle priorità multiple in modo automatico ed eseguirà anche automaticamente la vettorializzazione degli interrupt o *interrupt-vectoring:* ogni volta che un livello di interrupt è stato rivelato e accettato, esso causerà un branch del programma ad un indirizzo specifico nella memoria ove risiede la routine di servizio dell'interrupt.

Controllore di Accesso Diretto alla Memoria (DMAC)

La velocità di trasferimento in parallelo dei microprocessori monolitici è limitata dalla programmazione associata. Per trasferire una parola o una catena di caratteri devono venir eseguite diverse istruzioni. Questo vincolo può diventare critico poichè potrebbe rendersi impossibile gestire trasferimenti di blocchi di dati ad una velocità adeguata ai floppy disk o ai CRT. Un DMAC realizza in hardware gli algoritmi di trasferimento. Esso automatizza i trasferimenti di parole o blocchi tra la memoria e un dispositivo di I/O. Verrà descritto in dettaglio nel seguito.

CONTROLLORI DI PERIFERICHE

È possibile assiemare una CPU completa di memoria con un piccolo numero di chip. Tuttavia, fino a poco tempo fa, era necessario usare un gran numero di logica SSI ed MSI per interfacciare le periferiche al sistema base. Quest'ultima incongruenza è stata ora risolta. Esiste ora in un unico chip l'interfaccia verso numerosi dispositivi standard. In più esistono per questi dispositivi le logiche di controllo necessarie contenute in un unico chip: essi sono i nuovi controllori di periferiche. Esistono oggi controllori a chip singolo per un disco, una tastiera, un display, una stampante, un CRT ed altro ancora. La complessità di tali dispositivi è di solito analoga alla complessità di un microprocessore. Ad esempio un controllore per un disco si poteva realizzare con una scheda intera di logica. Esso è ora integrato su un singolo chip. Ne risulta che ora un sistema completo può essere costituito quasi esclusivamente da componenti LSI.

Se richiedeste una o più schede di SSI/MSI per un progetto di interfaccia standard, il vostro progetto potrebbe essere già obsoleto!

NUOVE COMBINAZIONI

Un vantaggio importante della maggior parte dei componenti dianzi indicati è l'aumentata efficienza del sistema. Essi lasciano libero il microprocessore per altri compiti. Lo svantaggio è l'introduzione di un chip in più nel sistema. Al fine di minimizzare questo svantaggio si stanno introducendo nuovi chip che offrano una combinazione delle funzioni anzidette. Via via che l'integrazione di componenti su un singolo chip aumenta, la dimensione richiesta per realizzare tutte le diverse funzioni descresce e diventa possibile costruire dei chip combinati. In particolare esistono ora chip combinati di memoria e di I/O. Ad esempio l'8085 può realizzare un sistema completo con soli tre chip usando chip combinati di memoria e di I/O.

Similmente esistono altri chip combinati per le funzioni di input-output: conversione da serie a parallelo più interfaccia parallelo, conversione da analogico in digitale più da serie a parallelo.

In futuro potrebbero venir realizzate ragionevolmente su un chip unico quasi tutte le combinazioni. In particolare ci si potrebbe aspettare presto l'avvento di un microprocessore, completo al suo interno di una conversione analogica/digitale per uso

industriale. Essi sarebbero ancora microprocessori digitali, ma avrebbero la capacità di trattare direttamente segnali analogici sull'ingresso o l'uscita.

Come ulteriore esempio, oggi un timer è quasi sempre contenuto direttamente entro lo stesso microprocessore. Si potrebbero anche associare timer addizionali entro ciascun chip combinato di memoria più I/O.

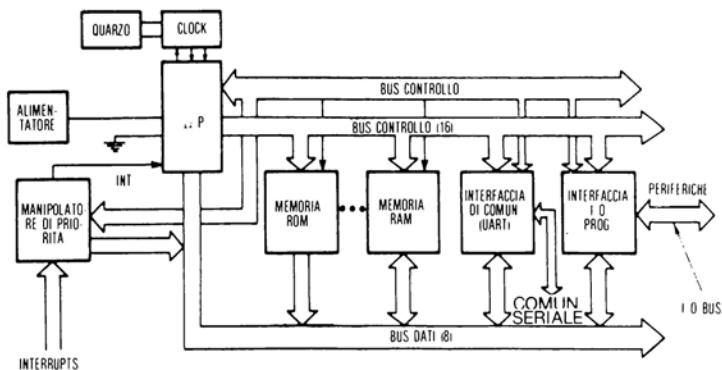



Fig. 3-1: Architettura di un sistema microprocessore standard.

Interconnessioni

La Fig. 3-1 mostra l'architettura di un sistema standard a microprocessore. La MPU sulla sinistra crea i tre bus del sistema. Tutti i chip richiesti sono connessi a questi bus. I tre chip fondamentali che sono necessari per il funzionamento del sistema, la ROM, la RAM e l'interfaccia per l'I/O, compaiono sull'illustrazione connessi ai bus. I componenti che verranno collegati al nostro sistema verranno ora studiati in dettaglio. Essi saranno collegati ai bus al Capitolo 5.

LA MEMORIA

Nei sistemi a microprocessore sono usate due principali tecnologie di memoria. La RAM è una memoria di lettura/scrittura. I suoi contenuti possono essere scritti o letti. Lo svantaggio fondamentale di una RAM sta nel fatto che essa è *volatile:* ogni volta che l'alimentazione viene a mancare, i contenuti della RAM vanno perduti. Per questa ragione il programma raramente risiede in una RAM. Se l'alimentazione dovesse mancare, sarebbe necessario ricaricare tutti i programmi prima di poter riprendere l'esecuzione. Le RAM sono perciò usate essenzialmente per memorizzare *dati,* come valori misurati o i risultati di calcoli intermedi, la cui perdita non sia critica in caso di un guasto all'alimentazione.

Il secondo tipo di memoria è la ROM, o memoria a sola lettura. Una volta che i contenuti di questa memoria siano stati definiti da un processo di fabbricazione, es-

si non potranno mai più essere alterati. Essi potranno essere letti, ma nessun contenuto potrà mai essere scritto. Una ROM perciò è utilizzata per immagazzinare i *programmi*. Essa è non volatile.
Esistono diversi tipi di memorie e verranno esaminate nei prossimi paragrafi.

RAM

RAM deriva dalle iniziali di random-access-memory o memoria ad accesso casuale. Di fatto, sia le ROM che le RAM sono memorie ad «accesso casuale». Questo è semplicemente il nome usato per tradizione per la memoria di lettura/scrittura. Per le memorie RAM si usano due tecnologie: la RAM statica e la RAM dinamica.

Una RAM *statica* immagazzina il singolo bit di informazione entro un flip-flop. Essa è asincrona e non richiede un clock. I suoi contenuti rimangono stabili permanentemente fintanto che vi è una alimentazione disponibile.

Una RAM *dinamica* immagazzina il singolo bit di informazione sotto forma di una carica. Una RAM dinamica utilizza la capacità gate-substrato del transistore MOS come una cella di memoria elementare. Il vantaggio ovvio è che questa cella elementare è più piccola di un flip flop di RAM statica. Ciò conduce ad una densità molto più alta. Oggi è possibile realizzare una RAM dinamica a 16K bit su un singolo chip. Presto sarà disponibile una RAM dinamica a 64K bit.

Inoltre la geometria più semplice della cella elementare porta come conseguenza una più alta velocità. La velocità tipica di una memoria RAM dinamica è compresa oggi tra i 300 e i 600 ns.

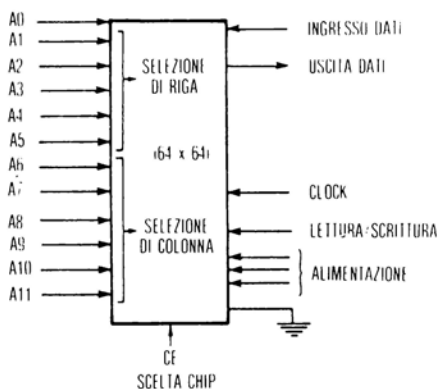



Fig. 3-2: RAM dinamica tipica da 4K.

Lo svantaggio della RAM dinamica è che la carica immagazzinata nel condensatore, come qualsiasi carica, è soggetta a dispersioni. Entro pochi millisecondi la maggior parte della carica va perduta. Onde preservare l'informazione contenuta in

una memoria RAM dinamica, è necessario *rinfrescarla* ogni 1 o 2 millisecondi. Il procedimento di refresh (rinfresco) consiste nella lettura e prelievo dell'informazione al di fuori della memoria e nella riscrittura nella medesima posizione. Per risparmiare tempo il procedimento di refresh preleverà una riga o una colonna completa per volta. Va notato che l'organizzazione interna della memoria non corrisponde al suo aspetto esterno. In particolare una tipica RAM dinamica da 4K, illustrata in Fig. 3-2, potrebbe avere 64 colonne x 64 righe. Perciò per il refresh completo di questa memoria saranno necessarie solamente 64 operazioni. La necessità di fornire un refresh presenta due svantaggi. Il primo consiste nella necessità di realizzare una logica di refresh sulla medesima scheda. In futuro questa logica potrebbe essere realizzata direttamente sul chip. Il secondo svantaggio sta nella possibilità che il refresh interferisca con il tempo d'esecuzione del processore durante l'esecuzione del refresh. Tuttavia si può facilmente dimostrare che l'impatto del refresh è dell'ordine dall'1 al 5 per cento. Esistono diverse tecniche di refresh che consentono un tempo di rallentamento limitato. La degradazione di velocità introdotta dalla logica di refresh non è perciò sostanziale.

Esempio: la Intel 2107B, che può essere considerata uno standard industriale, ha un tempo d'accesso di 200 ns (tempo necessario per leggere e prelevare un dato sull'uscita) e un tempo di ciclo di 400 ns (tempo necessario per scrivere dati in memoria), con 22 pin; ciascuna cella richiede un singolo transistore. Questa memoria è organizzata come 4Kx1 bit. Essa richiede tre livelli di alimentazione: + 5V, − 5V, + 12V.

Per realizzare una memoria di 4K parole utilizzando la 2107B, sarà necessario assiemare otto chip di questo tipo in parallelo. Ciascun bit costituente il dato di ciascun chip sarà collegato ad una delle linee del data bus. Queste RAM standard sono le più economiche. Per sistemi più piccoli sono disponibili architetture diverse, come le 1Kx4, o le 512x8 bit. Esse richiederanno meno componenti nel caso di una piccola memoria, ma saranno più costose.

La statica paragonata alla dinamica

Un circuito di RAM dinamica è più economico di quello di una RAM statica a causa della maggior densità ottenibile. Tuttavia una RAM dinamica richiede in più un circuito di refresh, per lo meno fino a quando questo non sarà incorporato entro l'MPU stessa. Per un *piccolo* sistema una RAM statica sarà generalmente più economica. Con ciò naturalmente si assume che non vi siano previsioni di espanderla in un tempo successivo fino a costituire un sistema di memoria più grande. Per un sistema *medio* o *grande*, la RAM dinamica è la più economica.

Lettura da una memoria

Per leggere o scrivere da o entro un chip di memoria è necessario:

1. Comunicare al dispositivo che ci stiamo indirizzando ad esso. Questo è il segnale di chip-select (CS) o chip-enable (CE). Esso seleziona un chip di memoria tra i

diversi chip collegati all'address bus.

2. Fornire l'indirizzo della parola scelta entro la memoria. Questo indirizzo sarà presentato ai pin di indirizzo del dispositivo. Ad esempio nel caso della RAM da 4K illustrata in Fig. 3-2, sarà necessario fornire un chip-select per selezionare un componente e 16 bit su A0-15 per fornire i 16 bit di indirizzo, che selezioneranno una parola tra le 4K disponibili.

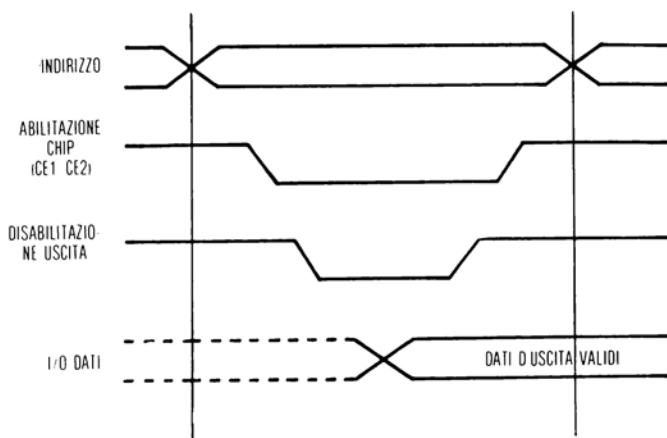



Fig. 3-3: Memoria statica: LETTURA.

Dopo un tempo detto «access-time» o tempo di accesso, il dato diverrà disponibile e apparirà sui pin dei dati. Esso verrà allora trasmesso lungo il data bus al microprocessore.

Nel caso di una memoria statica, i contenuti del data bus devono venir convalidati quando i dati vengono presentati. Questo compito viene svolto dal segnale di enable in Fig. 3-3. La transizione da alto a basso del segnale indica che il dato presentato nel data bus si presume valido.

L'indirizzo deve essere mantenuto valido almeno per la durata dell'access-time e, usualmente, più a lungo onde prevenire condizioni di criticità sui ritardi. Il segnale di *chip-select* o *chip-enable* viene di solito presentato contemporaneamente al resto dell'address-bus, ma non necessariamente. Il tempo di accesso è misurato dal momento in cui il segnale di chip-enable diventa operativo, cioé da quando l'accesso vero e proprio ha inizio. L'*access-time* è il tempo che intercorre tra la presentazione del segnale CS e la disponibilità di dati validi sui pin di uscita. Dopo ciascun ciclo di lettura, la memoria completerà un ciclo di scrittura, in modo che essa non potrà essere utilizzata di nuovo prima del termine del suo completo *tempo di ciclo* o *cycle-time*.

Scrittura in memoria

La sequenza è analoga all'operazione di lettura che è stata appena descritta. Il dispositivo deve essere *selezionato* e la parola entro la memoria deve essere specificata mediante i pin di indirizzo. Inoltre i dati che devono essere scritti nella memoria devono venir presentati sui pin dei dati entro un tempo specificato, T1. Dopo un tempo T2 chiamato tempo di ciclo, i dati saranno stati scritti all'indirizzo di memoria specificato. Sia il data bus che l'address bus saranno di nuovo disponibili. Il diagramma di temporizzazione (o delle fasi) nel caso di una memoria statica compare in Fig. 3-4.

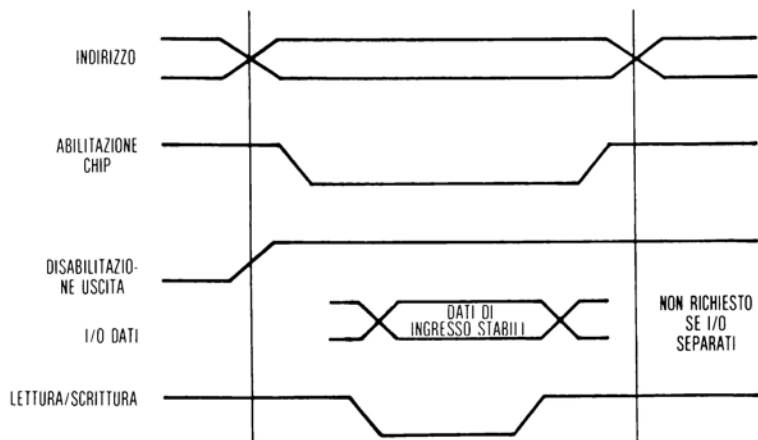



Fig. 3-4: Memoria statica: SCRITTURA.

Ciclo di lettura - modifica - scrittura

Questo ciclo speciale di read-modify-write offre dei vantaggi importanti nel trattamento delle liste (file), quando l'utilizzatore legge dapprima un byte, e quindi lo riscrive ancora nella medesima locazione di memoria.

Il problema della volatilità della memoria

Il problema principale delle RAM MOS è la *volatilità*. Qualora l'alimentazione si guastasse i contenuti della RAM svanirebbero. Questo è il motivo per il quale si preferiscono memorie a sola lettura (ROM) per memorizzare i programmi del microprocessore. Tuttavia in alcune applicazioni si devono preservare persino i dati memorizzati nella RAM. Inoltre è spesso desiderabile caricare programmi differenti in memoria in funzione di condizioni esterne (come ad esempio l'orario della giornata). Se si dovessero usare per queste applicazioni delle ROM, tutti i programmi

dovrebbero risiedere entro queste ROM conducendo ad una memoria MOS molto grande. Per entrambe queste ragioni è desiderabile disporre di una RAM non volatile.

Esistono essenzialmente due soluzioni per questo problema:

1. *Batteria in Back - up*

La soluzione più semplice e più frequentemente usata è di provvedere delle batterie per l'alimentazione della RAM durante guasti temporanei dell'alimentazione. Purché venga utilizzata una memoria CMOS, il consumo di potenza è bassissimo. Sono ora disponibili, derivate dall'esperienza aerospaziale, nuove batterie in un volume molto piccolo che potranno alimentare una scheda di memoria da 4K per settimane. Con batterie più ingrombranti e costose, la scheda di memoria verrà alimentata autonomamente per mesi. Ciò naturalmente implica che non solo la memoria, ma anche i decodificatori e la circuiteria di refresh dovranno essere anch'essi CMOS. Questa soluzione è adottata frequentemente nei sistemi portatili o nei sistemi industriali dove è altamente desiderabile preservare i dati durante brevi interruzioni dell'alimentazione che perdurino da pochi millisecondi a minuti. Il costo associato al ripristino di un processo industriale è generalmente così alto da giustificare il costo addizionale per fornire questo back-up (o sostegno in condizioni limite) mediante una batteria. Usualmente non è necessario fornire un lungo periodo di sopravvivenza con la batteria. Nell'eventualità di una mancanza prolungata dell'alimentazione si assume che il processo da controllare si arresti comunque in modo che non è essenziale fornire una sopravvivenza di lunga durata. Generalmente è essenziale invece una sopravvivenza di breve durata.

2. *Le EAROM*

Il termine EAROM deriva da Electrically - Alterable - Read - Only - Memory, cioè ROM alterabile elettricamente. Essa è infatti una memoria a lettura/scrittura o, più esattamente, una «read - mostly», cioè principalmente a lettura. Questo tipo di memoria verrà descritto nel prossimo paragrafo. In breve essa è costosa, molto lenta nelle operazioni di scrittura ed è stata usata per applicazioni particolarmente critiche come sistemi di guida per missili.

LE READ - ONLY MEMORIES (ROM)

Una read-only memory è una memoria che, una volta «programmata», può solamente essere letta dall'MPU. «Programmazione» qui significa che la configurazione di bit specificata è stata depositata nella memoria. Una memoria a sola lettura è intrinsecamente *non volatile.* Essa è quasi sempre usata per memorizzare i *programmi.*

Oggi esistono tre tipi principali di ROM: la ROM pura e semplice, la PROM, la EPROM (o RPROM) e la EAROM.

UNA ROM è una memoria a sola lettura programmata mediante mascheratura. La configurazione di bit corrispondente alla «programmazione» desiderata di questa memoria deve essere fornita dall'utilizzatore in un formato standard. Gli «0» e gli «1» verranno realizzati sulla memoria stabilendo o non stabilendo delle connessioni tra le righe e le colonne. L'ultimo passo di fabbricazione di un chip ROM è la fase di metallizzazione, nel quale vengono stabilite queste interconnessioni. Una volta che il costruttore abbia fornito la configurazione di bit, il costruttore potrà realizzare una maschera per la fase di metalizzazione e può eseguire quest'ultimo passo di fabbricazione.

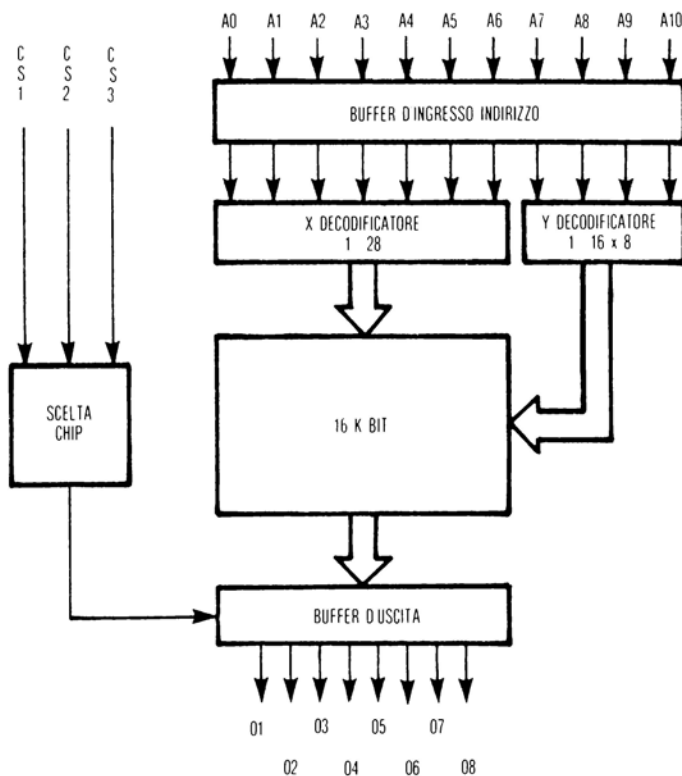



Fig. 3-5: Una ROM statica da 16 K (8316A).

Considerando il costo legato alla produzione di una maschera e, nel processo di fabbricazione, il costruttore normalmente richiede un quantitativo minimo nell'ordine prima di accettare questo compito. Tipicamente si devono produrre almeno 1000 ROM in una volta. Inoltre esiste un ritardo tipico da tre a sei settimane per la produzione. *Le ROM mascherate sono adatte unicamente ai grandi volumi di pro-*

duzione. I loro vantaggi sono molti: alta densità di bit, non volatilità e un costo più basso di ogni altro tipo di memoria, purché in grandi quantitativi.

Ad esempio la Fig. 3-5 mostra la struttura interna di una ROM statica a 16K, la 8316A. Essa è organizzata come una 2048x8 bit. Il suo tempo di accesso è al massimo di 850 ns.

Il tipico diagramma delle fasi per prelevare un dato letto su una ROM compare in Fig. 3-6. L'indirizzo deve mantenersi valido durante tutto il tempo di ciclo della

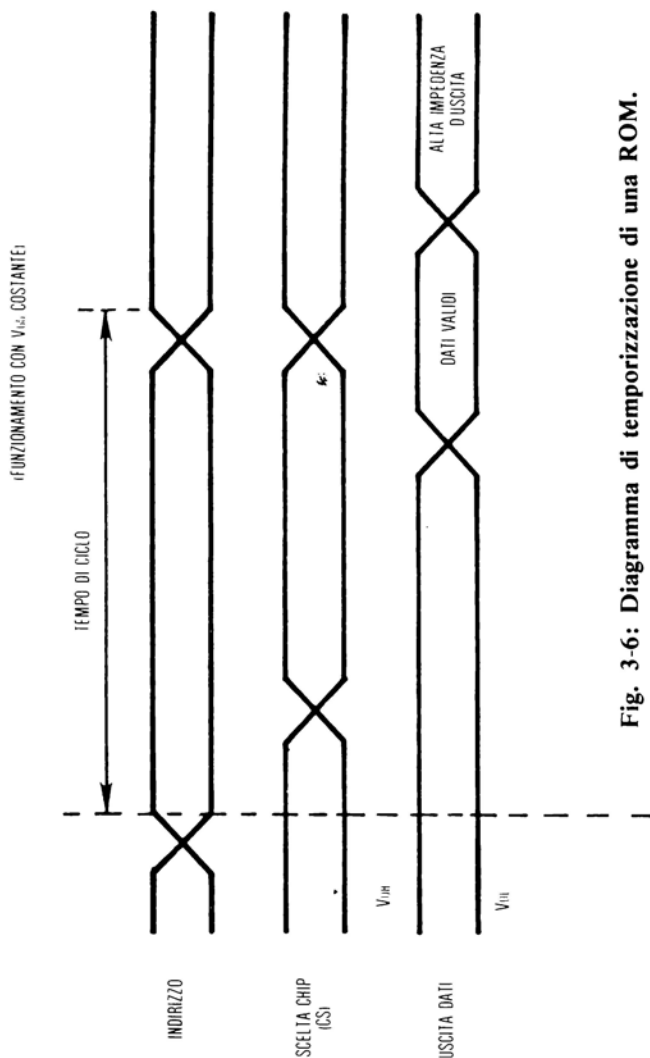



Fig. 3-6: Diagramma di temporizzazione di una ROM.

memoria e il chip select determina l'istante a partire dal quale la memoria inizierà il ciclo. Alla fine del tempo di accesso (la cui durata è ovviamente inferiore al tempo di ciclo), i dati sono presenti e validi sui pin di uscita.

I due svantaggi principali di una memoria ROM sono il ritardo associato alla sua produzione e il fatto che se ne debbano produrre un grande quantitativo. Inoltre le ROM non possono venir modificate una volta che esse siano state fabbricate. Se si dovesse riscontrare un errore nel programma sarebbe impossibile effettuare un cambiamento entro la ROM. Esse devono venir completamente rimpiazzate. Queste limitazioni sono esattamente contrarie alle esigenze di una fase di sviluppo per qualsiasi sistema e a quelle della produzione di sistemi in numero limitato. Per questa ragione sono stati introdotti diversi altri tipi di memorie a sola lettura. Tutti questi tipi possono essere «programmati» direttamente dall'utilizzatore. Si possono distinguere tre tipi principali.

PROM (Read Only Memory programmabili dall'utilizzatore)

Questa è una memoria a sola lettura che può essere programmata direttamente dall'utilizzatore, usando uno speciale programmatore per PROM. Essa è chiamata anche PROM fusible-link. Ciascuna cella di memoria è realizzata con un fusibile. La connessione fusibile (fusible link) può essere nichel cromo o silicio poly (cioé silicio policristallino). Durante il processo di programmazione, il programmatore PROM genererà dei treni di impulsi. I fusibili selezionati entro la memoria verranno fusi, permettendo alle giunzioni riga-colonna di essere aperte in queste posizioni. L'espressione colloquiale è di «bruciare» una PROM. Le PROM offrono un'alta densità, alta velocità e relativamente basso costo. Entro un tempo di pochi minuti è possibile bruciare una piccola PROM e inserirla nella sua applicazione. Inoltre la maggior parte delle PROM è direttamente compatibile con la ROM che potrà sostituirla in un futuro. Le PROM presentano le medesime caratteristiche di velocità e i medesimi collegamenti ai piedini. Questo naturalmente è un vantaggio serio per i prototipi di sistemi. Non è necessario bruciare tutte le locazioni di PROM in un'unica operazione. Tipicamente verrà programmata solo una parte di PROM. Qualora venissero individuati dagli errori di programma, sarebbe possibile usare la parte restante di questa PROM per cambiamenti e correzioni. Oppure si può gettare via la PROM e programmarne una nuova.

Pochi anni fa le PROM erano affette da un problema di affidabilità: una piccola percentuale delle PROM era soggetta al problema della «ricostruzione» o growback. Una parte del nichel e del cromo che si era volatilizzato dal fusibile poteva migrare dopo un po' di tempo e ristabilire così un percorso conduttivo. Perciò, un bit che era stato programmato a 0 poteva improvvisamente tramutarsi ancora in un 1. Il problema, tuttavia, da allora è stato risolto. Oggi vengono utilizzate nuove leghe e il problema si considera risolto. Come testimonianza dell'affidabilità delle nuove PROM si può evidenziare che il laboratorio del Mars Lander (n.d.t. veicolo spaziale fatto atterrare su Marte) che ha eseguito prove biochimiche per indagare

sull'esistenza di forme di vita, era programmato con delle PROM.

La maggior parte delle applicazioni che non verranno prodotte in grande serie, come molte applicazioni mediche, di laboratorio, industriali, aeronautiche o militari, utilizzano delle PROM come mezzo di immagazzinamento permanente per il programma. Nella maggior parte dei sistemi che devono venir prodotti in quantità di decine o centinaia, si dovrebbero considerare le PROM come il mezzo standard di immagazzinamento dei programmi. Nei sistemi che devono essere prodotti in quantità di migliaia e oltre, le ROM sono il normale mezzo di immagazzinamento.

Permane uno svantaggio delle ROM e delle PROM. Entrambe, una volta programmate, non possono venir cambiate. Durante la fase iniziale di sviluppo di un nuovo prodotto, è normale aspettarsi numerose variazioni del programma. Considerare le PROM in una tale fase iniziale sarebbe costoso, costituirebbe uno spreco e si rivelerebbe non pratico. L'alternativa è la EPROM:

La EPROM/RPROM (Erasable PROM o Ri-Programmabile PROM)

Queste sono memorie a sola lettura programmabili dall'utilizzatore che possono essere riprogrammate un certo numero di volte. Ne esistono di due varietà principali: esse sono rispettivamente le PROM cancellabili con gli UV e le PROM cancellabili elettricamente. Esse sono chiamate EPROM o RPROM secondo i gusti del costruttore. Il principio è il medesimo. Una EPROM tipica si cancellerà esponendola a luce ultravioletta dura da cinque a dieci minuti (n.d.t. «dura» sta a significare verso l'estremo alto della banda UV). I contenuti di tutte le celle di memoria vengono riportate a 0 (esse vengono scaricate). Il package di una EPROM ha un aspetto caratteristico: il coperchietto che sigilla il chip non è opaco, ma è una finestrella di quarzo che permette il passaggio degli ultravioletti. Dopo essere stata azzerata, la EPROM può essere «programmata» con uno speciale programmatore di (E) PROM. Locazioni scelte entro la EPROM possono allora essere programmate e una configurazione di bit può essere posta nella EPROM in pochi minuti. Il componente può allora essere inserito sulla scheda ove deve essere applicato. Se si scoprono degli errori o si desiderano cambiamenti, esso può venir scollegato e riprogrammato in pochi minuti. Questo procedimento può essere ripetuto un gran numero di volte (decine o centinaia).

Tuttavia le EPROM sono costose (35$ è un prezzo tipico per una 16K). Inoltre esse non sono generalmente compatibili pin per pin con la ROM o la PROM che verrà posta nel sistema. Le rispettive velocità e densità sono anch'esse significativamente differenti. Una scheda che utilizzi una EPROM richiederà quindi normalmente delle variazioni di progetto nell'indirizzamento e nella disposizione per il progetto finale che preveda ROM o PROM.

Per realizzare le EPROM si usano diverse tecnologie. La tecnica del «gate flottante» che è frequentemente utilizzata, accumula una carica in un gate di silicio poly «flottante» sopra il substrato di silicio, ma isolato da esso da uno strato di biossido

di silicio. La carica viene indotta nel «silicon gate» da treni di impulsi. Una volta programmata una EPROM riterrà la sua carica per diversi anni. Ci si aspetta che ritenga la sua carica per 10 anni con una perdita solamente del 30%. La cancellatura della carica viene realizzata facilmente: con luce ultravioletta dura, i fotoni incidenti sul silicon gate flottante ecciteranno gli elettroni dai livelli energetici più deboli e li faranno migrare verso il substrato di silicio dove la loro carica sarà neutralizzata. Questo è un processo essenzialmente analogo a quello fotoelettrico. Quando una carica è neutralizzata, il bit corrispondente viene ricondotto a 0. Una tipica EPROM da 8K-bit (la 8708) compare in Fig. 3-7. Il suo tempo di accesso è di 450 ns.

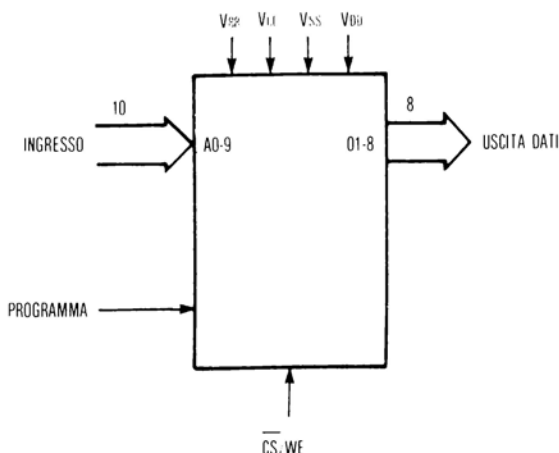



Fig. 3-7: EPROM tipica da 8 bit (8708).

Sono disponibili altri tipi di RPROM che sono direttamente cancellabili elettricamente. Essi stanno iniziando a competere con le EPROM a UV sul mercato ed ottengono caratteristiche simili.

EAROM (Elettricamente Alterabile ROM)

Un nuovo tipo di ROM è apparso sul mercato fin dal 1976. La EAROM può essere letta e scritta. Tuttavia i due procedimenti sono significativamente differenti. Essa dovrebbe venir qualificata come memoria «principalmente di lettura». Nella EAROM di fatto si può scrivere. Tuttavia l'operazione di scrittura richiede un *millisecondo*, mentre l'operazione di lettura può essere realizzata in un *microsecondo*. Essa non può essere usata come una memoria di impiego generale di lettura/scrittura. Inoltre usa tecnologie complesse che determinano basse densità e la necessità di alimentazioni multiple. La sua applicazione è limitata ad applicazioni critiche in-

dustriali e militari. Tipicamente una EAROM verrà usata nei casi ove sia necessario memorizzare un piccolo numero di parametri di quando in quando, ma abbastanza raramente. Il suo vantaggio è di essere non volatile e di non richiedere un'alimentazione ausiliaria come una batteria. Ad esempio può immagazzinare i parametri di pilotaggio nel caso di un sistema di pilotaggio automatico per un missile. Questa memoria è stata concepita per essere usata come una ROM e, di quando in quando, come una unità di memorizzazione non volatile. Il tempo di lettura per queste memorie è talmente basso che una loro vasta applicazione in sistemi tradizionali a microprocessore non può venire per ora considerata. Il progetto delle EAROM si sta evolvendo verso alte densità, maggiori velocità di scrittura e più bassi costi. Esse potrebbero diventare utilizzabili per applicazioni di uso generale entro pochi anni.

Tecnologie ROM

Per realizzare le ROM descritte vengono usate tre tecnologie principali:

1. *Bipolare*

La bipolare è utilizzata per le PROM cioé per le memorie programmabili dall'utilizzatore. Esse forniscono le più alte velocità: il tempo di accesso può essere inferiore ai 100 ns. Ciò permette velocità e corrispondenza sulla piedinatura compatibile con le ROM. Per le PROM più piccole si può ridurre il tempo di accesso fino a 40 ns. o anche meno.

2. *MOS*

La tecnologia MOS è usata per le PROM cancellabili, cioè le EPROM e le RPROM. Esse sono più lente delle precedenti: il tempo tipico di accesso è di 450 ÷ 1200 ns. La densità tipica va da 8K a 16K bit. Il costo di un dispositivo di programmazione (programma) per EPROM va da poche centinaia di dollari a $ 2.000 in proporzione alle facilitazioni offerte.

3. *MNOS*

La MNOS è una tecnologia MOS che utilizza metallo-nitruro sul gate. Essa va attentamente distinta dall'NMOS (MOS a canale n.). La MNOS è una tecnologia di primaria importanza per la realizzazione di EAROM (ROM alterabili elettricamente). Le sue caratteristiche fondamentali sono state già esaminate: un tempo di accesso dell'ordine di un microsecondo e un tempo di cancellazione da 1 a 100 millisecondi. La densità tipica oggi va da 1K a 4K. I costruttori principali sono la GI, la Nitron (una divisione della Mc Donnell-Douglas) e la NCR negli USA; la Nippon Electric in Giappone.

Altri tipi di memorie

Si possono menzionare altri tre tipi meno usati di memorie:

1. *Memorie a Bolle*

Le memorie a bolle stanno iniziando a farsi strada sul mercato. Esse sono carat-

terizzate da una densità estremamente alta. Una tipica memoria a bolle integra oggi 64K bit e oltre. Tuttavia esse sono lente e per ora costose. Esse potranno rappresentare una seria concorrenza in futuro per i piccoli dischi.

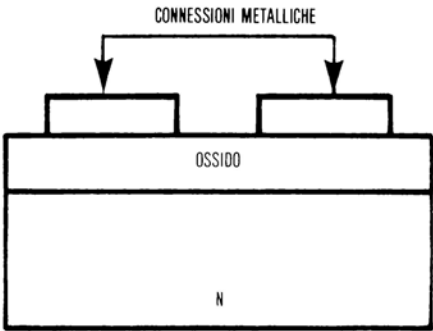



Fig. 3-8: Memoria CCD.

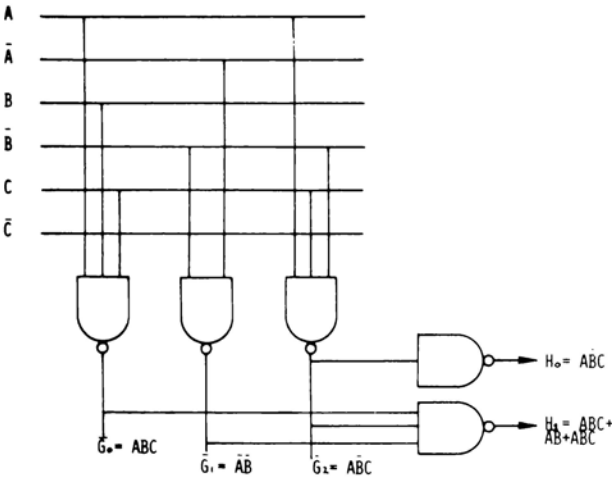



Fig. 3-9: Un PLA: diagramma logico.

2. CCD

CCD sta a significare charge-coupled-device o dispositivo ad accoppiamento di carica. Queste sono memorie MOS caratterizzate da una densità altissima. La struttura della memoria CCD compare in Fig. 3-8. Un certo numero di piccoli ret-

tangolini di alluminio vengono depositati sul silicio. Per merito della geometria semplicissima e altamente ripetitiva si possono ottenere altissime densità e sono oggi disponibili 64K bit su di un unico chip. Il tempo di accesso è ancora lento (da 50 a 100 microsecondi) e il costo è relativamente alto. Il mercato principale delle memorie CCD è in concorrenza diretta con i dischi di piccola o media dimensione. A causa di un abbassamento nei costi della meccanica dei terminali a dischi e della introduzione di un controllore per dischi su 1 chip, si ha l'impressione che la CCD non potrà essere in diretta concorrenza con i dischi prima di almeno due o tre anni.

3. *PLA (Programmable-Logic-Array o Matrice di Logica Programmabile)*
La PLA non è una semplice memoria. La struttura di una PLA appare in Fig. 3-9. Essa è essenzialmente una combinazione di due ROM denominate: la AND-ROM e la OR-ROM. È usata per decodificare segnali logici o per codificarli. Le PLA sono usate specificamente nel progetto delle unità di controllo. Esse sono usate raramente nel progetto di microprocessori monolitici (almeno al loro esterno).

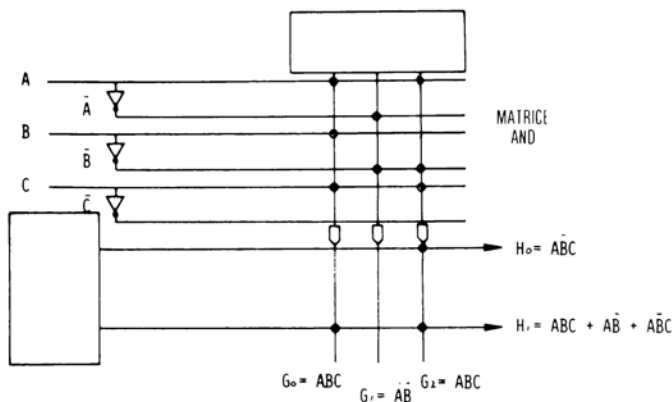



Fig. 3-10: PLA: matrice equivalente.

TECNICHE DI INPUT-OUTPUT

Il collegamento di un dispositivo di input-output generalmene richiede un'*interfaccia*. Il funzionamento del dispositivo può richiedere anche un *device-controller* (n.d.t. controllore del dispositivo). Infine l'utilizzo di una unità di input-output collegata ad un sistema di elaborazione richiede una *strategia di assegnazione di tempo* (scheduling).

Un'*interfaccia per il dispositivo* può variare in complessità da pochi registri o porte, ad una o più schede di logica. Per semplificare l'interfacciamento dei microprocessori sono stati sviluppati alcuni chip di interfaccia di impiego generale, che

saranno descritti in questo paragrafo. Le tecniche di interfacciamento effettive per i dispositivi connessi di input-output verranno descritte al Capitolo 7.

I *device-controller* sono richiesti di solito per i dispositivi che hanno parti meccaniche complesse. Il «controller» incorpora un'unità equivalente ad un processore che riceverà istruzioni e le eseguirà. Questo controller realizzerà la sequenza di controllo richiesta dal dispositivo. Ad esempio, farà avanzare un elemento meccanico, usando un motore passo-passo, azionandolo per un numero specificato di passi. I device controller possono differenziarsi da semplici dispositivi a realizzazioni estremamente complesse multi schede. I device controller sono oggi disponibili in forma LSI per i dispositivi più frequentemente usati. Alcuni esempi saranno presentati al Capitolo 7.

Una volta che un dispositivo di input-output sia stato collegato al sistema mediante un controller LSI e attraverso un'interfaccia LSI, si deve istituire una procedura di comunicazione tra il dispositivo e il processore. Sono state evidenziate tre essenziali *tecniche di scheduling* atte a controllare dispositivi di input-output. Queste tre tecniche saranno descritte ora.

Le tre tecniche sono illustrate in Fig. 3-11. Esse sono chiamate rispettivamente polling (o I/O programmato), interrupt e DMA.

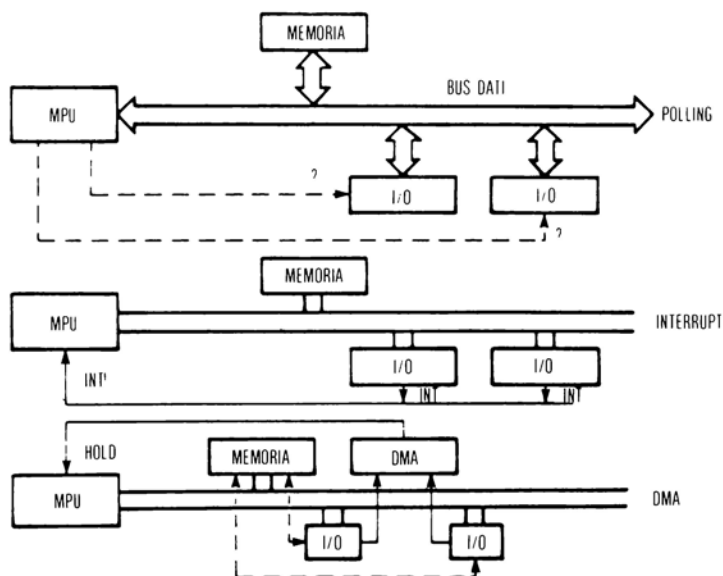



Fig. 3-11: Le 3 tecniche di I/O.

Polling

Il polling chiamato anche *I/O programmato* è la tecnica di scheduling più semplice. I dispositivi di I/O sono collegati semplicemente al data bus e all'address bus del sistema. In funzione del sistema a microprocessore essi devono anche essere collegati ad una combinazione di linee del control bus. Il principio di una tecnica di scheduling è di realizzare una procedura per stabilire quale sia il successivo dispositivo di input-output che richiede, o al quale verrà concesso il servizio. La tecnica di polling è chiamata tecnica *sincrona.*

Il microprocessore richiederà periodicamente a ciascun dispositivo collegato al proprio data-bus: «*Richiedi il servizio?*». Ciascun dispositivo risponderà allora con un «sì» o «no». Ogni volta che verrà ricevuto un «no» come risposta, il microprocessore procederà al dispositivo successivo e ripeterà la domanda «Richiedi il servizio?». Nella configurazione polling il microprocessore richiamerà ciascuno dei dispositivi di I/O successivamente e determinerà così se il servizio è richiesto oppure no.

In pratica verrà interrogata un flag sul dispositivo o sulla sua interfaccia. Se il test darà esito positivo, verrà intrapresa un'azione. Un'azione tipica è il trasferimento di una parola o di un blocco di dati verso oppure dal dispositivo.

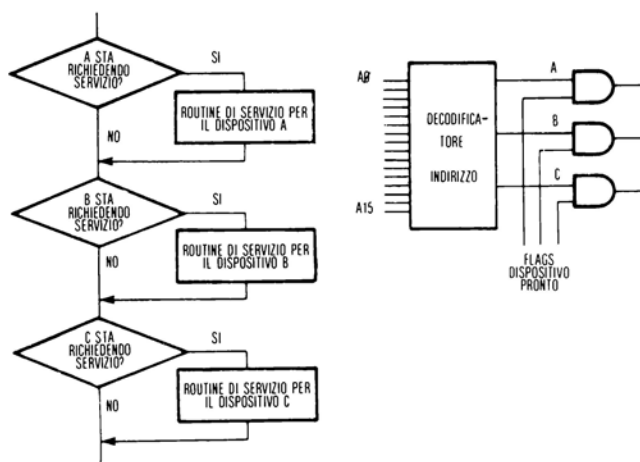



Fig. 3-12: Sequenza di polling.

Il programma usato per realizzare un algoritmo di polling è chiamato *polling loop.* La procedura di rivolgere un'interrogazione al dispositivo e di ricevere un'informazione di ritorno è chiamata «*handshaking*» (n.d.t.: stretta di mano). Ogni uno

tocollo di comunicazione tra un dispositivo e il successivo sul medesimo collegamento realizzerà normalmente qualche forma di «handshaking». In altre parole, prima di trasmettere informazioni ad un dispositivo, verrà verificato un bit di stato per vedere se questo dispositivo è pronto ad accettare i dati. Prima di leggere una parola di informazione dal registro di un dispositivo, si dovrà ancora verificare un bit di stato per vedere se questo registro è già aggiornato. La struttura concettuale del loop di programma compare in Fig. 3-12.

I vantaggi del polling sono diversi:

1. Richiede un hardware minimo per l'interfaccia stessa. Non sono richieste linee speciali.

2. Il suo vantaggio principale sta nel fatto che il polling è *sincrono* con l'esecuzione del programma. Si conosce esattamente quando un dispositivo verrà interrogato e quanto tempo impiegherà per essere servito. Non potrà capitare alcun evento che possa scompaginare la sequenza preordinata di polling. Per contrasto si dimostrerà che le altre due tecniche che descriveremo (interrupt e DMA) sono *asincrone.*

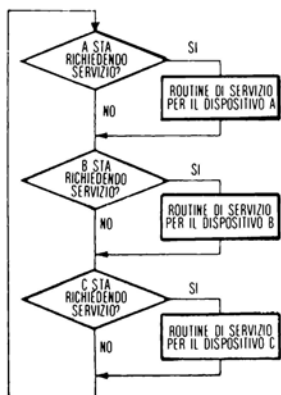



Fig. 3-13: Un ciclo di polling.

Lo svantaggio fondamentale del polling è la quantità di software che lo deve sostenere. Ogni volta che si arriva ad un loop di polling, tutti i dispositivi verranno verificati. In pratica la maggior parte di essi può non richiedere alcun servizio. Tuttavia, onde garantire che ciascun dispositivo venga verificato entro un periodo di tempo specificato, l'intero loop dovrà essere eseguito frequentemente anche se non se ne presenta alcuna necessità. Questo spreco di tempo di processore può sollevare obiezioni e in tale caso richiede l'uso di una delle altre due tecniche. Tuttavia ogni volta che questo «spreco» di tempo di elaborazione da parte del microprocesso-

re non solleva obiezioni (perchè non vi è niente altro da fare, cioè la velocità del microprocessore è sufficiente), questa è decisamente la tecnica più semplice da utilizzare. *La predicibilità dell'ordine con il quale i dispositivi verranno interpellati è un vantaggio fondamentale di programmazione.* Per questa ragione il polling viene normalmente consigliato a tutti quei programmatori che non siano già esperti nella progettazione condotta per interrupt. Si può persino supporre che se è necessaria una complessa struttura di interrupt in un sistema con microprocessore, si dovrebbe considerare un diverso approccio del progetto.

Interrupt

Ogni volta che il polling non consente un tempo di risposta sufficientemente rapido oppure spreca una porzione eccessiva del tempo concesso al microprocessore, si devono considerare configurazioni a interrupt. In uno schema di conduzione per interrupt i dispositivi periferici prendono l'iniziativa della richiesta di servizio. Si utilizza una linea in più che compare in fondo all'illustrazione in Fig. 3-14: questa è la linea di interrupt, connessa all'MPU. Ciascun dispositivo di I/O è ora collegato a questa linea di interrupt. La situazione è analoga a quella che si crea in un autobus. Ogni passeggero che desideri scendere ha la facoltà di inserirsi su una linea che azionerà un campanello (la linea di interrupt). Ciascun dispositivo che richieda il servizio può generare un impulso o livello di interrupt su questa linea. Il microprocessore dovrà *rilevare* la presenza di interrupt sulla linea e *regolarsi* di conseguenza. L'operazione dettagliata di interrupt verrà presentata nel prossimo capitolo sul

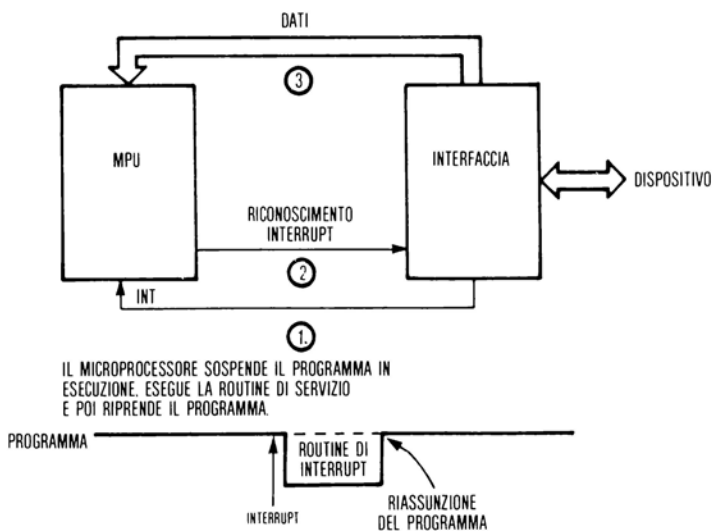



Fig. 3-14: Interrupts.

Coordinamento dei Chip di Interfaccia. In breve il microprocessore deve accettare l'interrupt, identificarlo e servirlo.

L'accettazione o il rifiuto di un interrupt vengono realizzati con un bit interno di mascheratura detto maschera di interrupt o inibizione di interrupt o bit di abilitazione di interrupt. Questo bit è normalmente memorizzato nel registro di flag o di stato.

Quando un interrupt è stato accettato, il microprocessore deve determinare quale sia il dispositivo che l'ha originato. Inoltre diversi dispositivi possono generare un interrupt simultaneamente. Per questa ragione ogni volta che alla medesima linea di interrupt siano collegati dispositivi multipli, si devono assegnare delle *priorità*. Questo problema verrà studiato nel paragrafo seguente.

Fig. 3-15: Richiesta contemporanea di servizio.

Assumendo ora che l'interrupt sia stato accettato e che il dispositivo sia stato identificato, deve essere svolto il servizio richiesto dal dispositivo. Il microprocessore deve sospendere il programma che stava eseguendo ed eseguire un branch ad una *routine di servizio interrupt* (o *interrupt handler*) per il dispositivo. Quando l'indirizzo richiesto per il branching è disponibile nel medesimo istante in cui l'interrupt viene presentato al microprocessore, esso è un «*interrupt - vettorizzato*».

L'esecuzione di una routine di interrupt è analoga a quanto avviene nel caso del polling. Non appena l'handler arriva al termine, il programma che era stato sospeso dall'interrupt deve venire riabilitato di nuovo sul processore. Ciò richiederà diverse istruzioni.

Il vantaggio essenziale degli interrupt è di fornire un responso rapido. Assumendo che non compaiano richieste in conflitto simultaneamente, i dispositivi riceveranno il servizio entro breve tempo dopo averlo richiesto. Questo «breve tempo» misura l'efficienza del microprocessore nel rispondere ad interrupts esterni. Gli interrupts sono generalmente richiesti nei sistemi *real-time*, che devono garantire il tempo di risposta migliore possibile alle condizioni esterne.

Gli svantaggi degli interrupts sono essenzialmente tre:

1. Richiedono dell'hardware in più, in particolare se delle priorità devono essere risolte esternamente (questo punto dovrà essere spiegato al prossimo paragrafo).

2. Si incorre in istruzioni aggiuntive ogni volta che un programma viene interrotto da un dispositivo esterno.

3. L'esecuzione dell'«handler» del dispositivo è ora asincrono rispetto all'esecuzione del programma principale. In linea di principio sarebbe desiderabile predire tutti i casi possibili in modo da poter anticipare le situazioni e riservare aree appropriate in memoria, oppure calcolare le temporizzazioni.

In pratica ciò può diventare estremamente complesso.

DOMANDA: *Guardando la Fig. 3-15 noterete che solamente una linea di interrupt viene usata per dispositivi di input-output. In pratica la maggior parte dei microprocessori adotteranno una o due linee di interrupt, molto raramente di più. Perché non adottare una linea di interrupt per dispositivo? (Questo è di fatto l'approccio scelto nei sistemi più grandi).*

La risposta dovrebbe ormai essere ovvia: sul microprocessore *non ci sono abbastanza pin* per adibire alcuna altra linea al control bus e in particolare a linee di interrupt. Per questa ragione la linea o linee di interrupt devono venir suddivise tra i vari dispositivi. Si vedrà come i risultanti conflitti di priorità vengono risolti sia mediante software, sia con hardware esterno (chip di priorità).

Gli interrupt possono non essere ancora abbastanza rapidi per alcuni dispositivi che richiedono trasferimenti veloci della parola. Ogni volta che viene ricevuto un interrupt, il microprocessore sospende il programma in esecuzione e commuta ad una nuova routine. Successivamente trasferirà di solito una parola. Lo farà mediante software. In previsione del numero di istruzioni che devono venir eseguite prima che una parola venga realmente trasferita, scorrerà generalmente un notevole numero di microsecondi. Questo tempo è troppo lungo e rende il sistema troppo lento per dispositivi come floppy-disk o terminali video con visualizzatore CRT che debbano essere rinfrescati. L'ultima soluzione per rendere il processo più veloce consiste nel realizzare direttamente ad hardware questo processo software che è stato identificato come collo di bottiglia. Questa è la tecnica DMA.

Direct - Memory - Access (DMA)

Un Controller di Accesso Diretto in Memoria (DMA o DMAC) è uno speciale *processor di trasferimento di blocchi.* Esso è un dispositivo hardware che realizza automaticamente e a velocità hardware, un processo che sarebbe normalmente eseguito da un programma entro il microprocessore. Il principio di funzionamento del DMAC appare in Fig. 3-16. Questa volta, invece di mandare un interrupt alla MPU, i dispositivi di I/O (o le loro interfacce) invieranno l'interrupt al DMA. Il DMA sospenderà allora il microprocessore inviandogli un segnale di «HOLD». Il medesimo DMA si assumerà allora il controllo del funzionamento del sistema e trasferirà automaticamente uno o più parole tra la memoria e il dispositivo di I/O. Il funzionamento dettagliato del DMA sarà spiegato nel prossimo paragrafo. Un DMAC è un dispositivo complesso: la sua complessità può essere comparata alla complessità di un microprocessore. Naturalmente esso è molto più costoso. Un

DMAC deve essere usato in applicazioni dove la velocità di trasferimento del blocco della MPU non è adeguato neppure con gli interrupt. Esso fornisce quindi trasferimenti ad alta velocità di parole o blocchi tra il dispositivo o la memoria. Un DMAC viene tipicamente usato nel caso di dispositivi veloci di I/O come dischi o CRT. Esso contribuisce significativamente al costo e alla complessità del sistema.

Tutto ciò riassume i principi fondamentali delle tre tecniche di sequenzializzazione dell'input-output. Ora verranno passati in rassegna i diversi dispositivi di interfaccia e verranno quindi usati per fornire i mezzi di input - output richiesti.

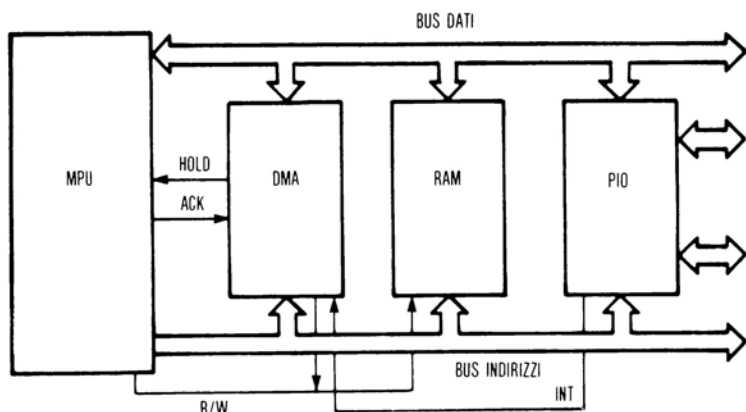



Fig. 3-16: Accesso diretto alla memoria (DMA).

CIRCUITI DI INTERFACCIA DI INPUT - OUTPUT

Verranno distinti qui tre tipi di chip: i semplici chip di interfaccia, i chip di controllo dispositivi (device-controller) e i chip di sequenzializzazione di interfaccia.

CHIP REGOLARI DI INTERFACCIA

Per le interfacce digitali si usano due tipi fondamentali di chip: l'interfaccia universale seriale e l'interfaccia parallela. Inoltre sono disponibili una gran varietà di chip convertitori analogico-digitale e digitale-analogico. Ora verranno esaminati in dettaglio questi due tipi essenziali di chip di interfaccia digitali. La comprensione di questi è fondamentale per la comprensione di un sistema a microprocessore.

L'UART

Un UART è un *Trasmettitore Ricevitore Asincrono Universale*. Esso converte un ingresso seriale in un'uscita parallela e può convertire simultaneamente un in-

gresso parallelo in un'uscita seriale. Il dispositivo asincrono è usato più frequentemente per funzionamento a bassa e media velocità. Un UART è un dispositivo *sincrono* usato per trasmissione ad alta velocità.

La funzione base di un UART è la conversione serie/parallelo. Il principio della conversione da serie a parallelo compare in Fig. 3-17. Sulla sinistra dell'UART è illustrato un segnale digitale come sequenza di 0 e di 1. Un «1» è un livello alto, mentre uno «0» è un livello basso. Si può vedere che il segnale di ingresso è (da destra a

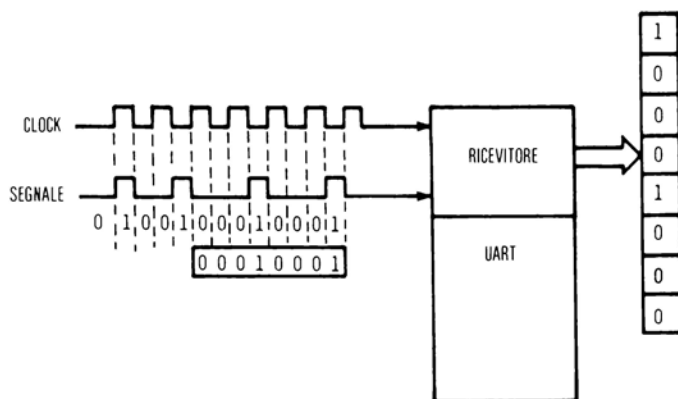



Fig. 3-17: Conversione seriale - parallelo.

sinistra) 1000100010010, ecc.. L'uscita del dispositivo appare sulla destra dell'UART: è 10001000. Quest'uscita deve essere predisposta su otto bit in parallelo. Sorge immediatamente un problema.

In questo esempio il primo «1» verrà individuato come «1». Tuttavia i successivi tre «0» non possono essere facilmente distinti. Come può sapere l'UART che ci sono tre zeri in successione piuttosto che uno? Evidentemente necessita di una *temporizzazione*. Questa temporizzazione viene fornita da un *clock* esterno, sincronizzato con il segnale esterno. Questo segnale di clock appare nella parte superiore sinistra della figura. Si deve notare che il clock esterno è sempre fornito dall'esterno e deve essere sincronizzato con il segnale serie. Un impulso del clock (tipicamente un mezzo periodo) identificherà la presenza (durata) di un bit. Tre successivi impulsi di clock delimiteranno e identificheranno allora i tre zeri in sequenza nel segnale di ingresso. Gli otto bit da assiemare come parola in parallelo sono mostrati immediatamente al di sotto del segnale di ingresso sulla sinistra di Fig. 3-17. Con questi due segnali di ingresso, clock e dati, la porzione ricevente dell'UART assiemerà automaticamente una parola di 8 bit e la renderà disponibile in uscita sulle sue connessioni a 8 bit con il data bus. Viceversa l'UART accetterà un segnale di ingresso di 8 bit (dal data bus del microprocessore) e lo serializzerà su una linea di uscita seriale.

sotto il controllo di un clock esterno fornito dal dispositivo esterno. Il ricevitore e il trasmettitore sono indipendenti.

Gli UART furono tra i primi chip LSI ad essere standardizzati. Come risultato la maggior parte degli UART oggi disponibili sono essenzialmente identici. Alcuni costruttori di microprocessori hanno aggiunto alcune prerogative in più oppure hanno riunito diversi dispositivi in uno solo per renderli differenti o «più adatti» al loro microprocessore. In breve, a parte la velocità, le loro caratteristiche sono essenzialmente analoghe.

Un UART standard ha tre sezioni: un ricevitore, un trasmettitore e una sezione di controllo. I tre moduli sono illustrati in Fig. 3-18.

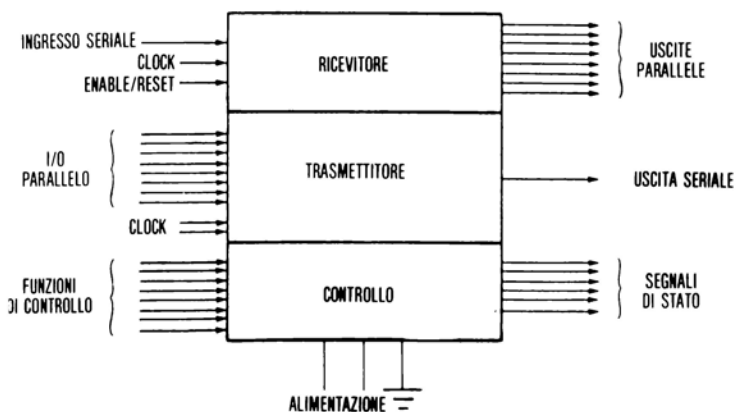



Fig. 3-18: Un UART standard è formato da 3 sezioni.

Il ricevitore riceve un ingresso seriale (più il clock) e pilota un'uscita di 8 bit in parallelo.

Il modulo trasmettitore riceve un ingresso parallelo di 8 bit (più il clock) e pilota un'uscita seriale.

Il modulo di controllo riceve l'informazione di controllo dal microprocessore e realizza le istruzioni richieste. Esso fornisce anche in uscita l'informazione di stato e controllo.

Oltre al processo di serializzazione/deserializzazione un UART fornisce un certo numero di funzioni standard. Controllerà automaticamente il bit di start, i bit di stop e verificherà automaticamente la corretta trasmissione dei dati usando, se richiesto, la verifica di parità.

Nella trasmissione standard a 8 bit, ciascun carattere di 8 bit è configurato ad esempio, con un bit di start e due bit di stop. La sezione di input estrarrà allora auto-

maticamente ciascun carattere dai suoi bit di start e di stop e riterrà solo i contenuti degli 8 bit. Similmente, durante la trasmissione, sommerà ai segnali in uscita i bit richiesti di start e di stop.

La verifica di parità è un bit in più usato spesso per verificare la trasmissione corretta dei dati. La parità aggiunge un bit in più ad un codice di 7 bit e serve a controllare il fatto che un singolo bit possa essere stato accidentalmente invertito. Il principio è il seguente: viene contato il numero degli «1» in una parola di 7 bit. Se questo numero è dispari e se usiamo uno schema di parità «pari», il bit di parità da aggiungere è «1». Se la parità fosse stata dispari, il bit di parità sarebbe stato «0». In altre parole in uno schema di parità «pari» il numero totale di 1 contenuti in qualsiasi parola di 8 bit viene garantito *pari*. Se qualsiasi singolo bit degli otto bit dovesse essere cambiato accidentalmente durante la trasmissione, uno speciale circuito di verifica di parità segnalerebbe questo fatto all'atto della ricezione. La verifica di parità scopre errori in singoli bit. Per segnalare errori su più bit esistono tecniche più sofisticate. Poiché lo schema descritto risolve la maggior parte dei problemi incontrati normalmente nella trasmissione, esso è il sistema più frequentemente usato. Un UART realizzerà automaticamente la verifica di parità o, se richiesto, la genererà ottenendo ciò mediante la parola di controllo appropriata. Esso realizzerà la parità pari o dispari oppure nessuna parità.

La lunghezza di una parola gestita dall'UART può essere di 5, 6, 7 oppure 8 bit. La parità può essere dispari o pari. La parità può esistere o meno. I bit di stop possono essere 1, 2 o 1 ½ (in un codice a 5 bit).

Le principali applicazioni dell'UART riguardano la comunicazione con dispositivi seriali come una telescrivente, una stampante o un modem (collegato ad una linea telefonica).

Esempio: L'Intel 8251 USART («PCI»)

USART 8251 INTEL

La 8251 dell'Intel è un UART. Inoltre è un USRT (Trasmettitore e Ricevitore Universale *Sincrono*); in altre parole può essere usato sia come dispositivo asincrono che come sincrono. Probabilmente per l'utilizzatore è di dubbio valore disporre di un dispositivo che possa essere usato in entrambe le modalità, sincrona o asincrona: in qualsiasi progetto normale il sistema sarà progettato per uno dei due sistemi di comunicazione o asincrono o sincrono. Non è certamente frequente che si possa cambiare questa scelta durante il progetto. Tuttavia essendo il prezzo dell'8251 sostanzialmente analogo alle altre UART sul mercato, la disponibilità di una modalità sincrona può essere considerata un vantaggio. L'organizzazione logica del dispositivo appare in Fig. 3-19. I tre blocchi funzionali di qualsiasi UART possono essere chiaramente identificati: la sezione trasmittente, la sezione ricevente e la sezione di controllo. Per maggior chiarezza è stato indicato un blocco separato,

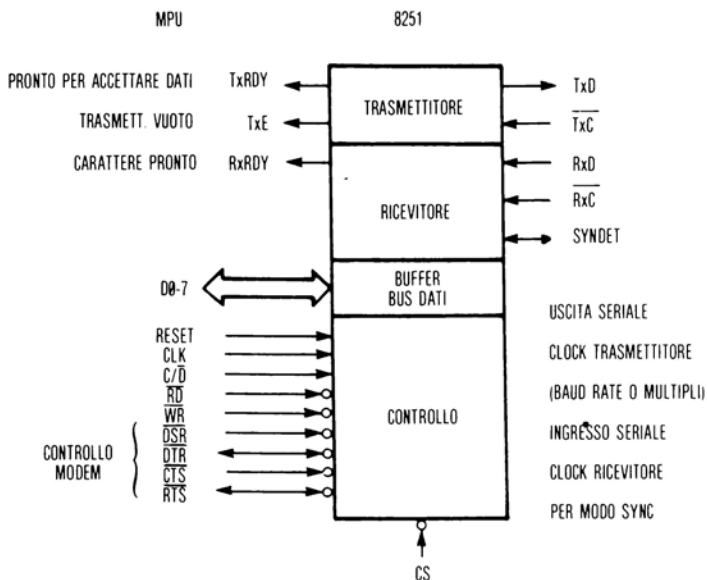



Fig. 3-19: USART 8251 della Intel.

definito «data bus buffer», che comunica con le altre zone. I collegamenti del dispositivo al microprocessore sono tutti sul lato sinistro della figura. I collegamenti del dispositivo alle periferiche sono tutti sul suo lato destro. Al solito ci sono due segnali per le funzioni di I/O: una linea dei dati e un segnale di clock. Inoltre vi è una linea di sincronizzazione per l'uso in modo sincrono, SYNDET. L'ingresso seriale è RxD; l'uscita seriale è TxD.

L'8251 viene selezionato da un segnale di abilitazione CS inserito dal basso in Fig. 3-19. Ogniqualvolta CS è 1, il dispositivo è abilitato. Esso può ricevere quattro ordini dal microprocessore che compaiono in Fig. 3-20. Questi ordini permetteran-

$\overline{C}$/D	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	FUNZIONAMENTO
0	0	1	0	8251 AL BUS DATI (LETTURA)
0	1	0	0	BUS DATI A 8251 (SCRITTURA)
1	0	1	0	STATO AL BUS DATI
1	1	0	0	BUS DATI A CONTROLLO
			1	BUS DATI A TRI-STATE

Fig. 3-20: Comandi USART.

no di leggere o scrivere sul buffer del bus dei dati della 8251, leggere il suo stato o scrivere una parola di controllo entro la sua sezione di controllo. Un'applicazione tipica dell'8251 nel modo asincrono appare in Fig. 3-21. In questo esempio l'UART viene usato per leggere l'informazione seriale da una tastiera e per visualizzare l'informazione su un CRT. Il generatore di «baud rate» (velocità di trasmissione di ciascuna parola seriale) che compare in figura è un modulo di ciascuna parola seriale) per fornire gli impulsi di clock adatti per la trasmissione. Ad esempio il modo più lento di funzionamento è normalmente 110 bauds (a tutti i fini pratici per una parola digitale binaria un baud corrisponde ad un bit al secondo, o bps. Ciò corrisponde a 10 caratteri al secondo, la velocità standard per le telescriventi). La maggior parte dei controllori per CRT è equipaggiata con un generatore di baud rate, dove il baud può essere selezionato da 110 bauds a 9600 bauds.

Nel funzionamento sincrono il dispositivo viene tipicamente collegato ad un modem per comunicare su una linea di dati ad alta velocità con un altro calcolatore come, ad esempio, un minicomputer. La trasmissione sincrona è caratterizzata dall'emissione di blocchi di dati ad alta velocità. In modo asincrono l'8251 arriverà fino a 9,6K bauds. In modo sincrono arriverà fino a 56K baud.

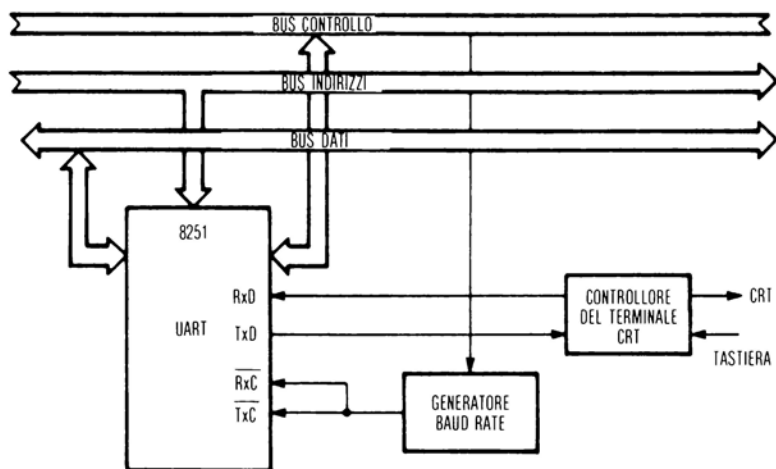



Fig. 3-21: Applicazione tipica della USART.

Il PIO

PIO significa chip di interfaccia «programmabile di input-output» oppure chip con presentazione «parallelo di input-output». La parola «PIO» non costituisce uno standard industriale. Nessuno standard è mai stato stabilito per questi dispositivi

nel modo in cui fu creato per gli UART. I costruttori utilizzano nomi diversi per dispositivi di questo tipo come «PIA» (Motorola), «PPI» (Intel), «PDC» (Rockwell), «PIO» (Zilog). Il termine «PIO» verrà usato nel seguito di questo libro per designare questo tipo di dispositivi. Il lettore dovrà semplicemente stare attento alle differenze che potrà incontrare nella letteratura di altri costruttori. Il PIO è un dispositivo di interfaccia *programmabile.* La sua funzione è quella di fornire un'interfaccia di base per 8 bit di dati in parallelo. Inoltre è programmabile. Entrambi i punti verranno ora chiariti.

Per collegare un dispositivo di input o di output al data bus di un microprocessore è necessario fornire normalmente un latch per l'input e un latch per l'output. Il latch di input manterrà i dati validi abbastanza a lungo da permettere al microprocessore di venirli a leggere. Esso isolerà altresì i segnali dal bus. Similmente un latch di uscita è necessario per congelare i dati in uscita abbastanza a lungo affinchè il dispositivo in uscita ne possa far uso. Ad esempio i dati presentati sul bus di un 8080 saranno validi tipicamente per meno di 500 nanosecondi. Questo tempo non è sufficientemente lungo da poter essere utilizzato dalla maggior parte dei dispositivi di input-output. Inoltre lo *stato* di questi registri deve essere disponibile per realizzare una procedura di «*handshaking*» nelle comunicazioni. Prima di leggere i contenuti di un buffer di ingresso, il microprocessore deve sapere che i contenuti di questo registro sono validi. Deve essere inviato al microprocessore un bit di stato o, in modo equivalente, un interrupt. D'altra parte un bit di stato deve denotare se il buffer di uscita è caricato o vuoto. Ciò è necessario affinchè il microprocessore possa determinare se può inviare sull'uscita la parola successiva. Ciò può essere necessario anche per i dispositivi di uscita per determinare se esso può usare i contenuti del buffer.

Riassumendo, qualunque interfaccia di I/O parallela deve fornire almeno un registro di ingresso, un registro di uscita e inoltre dei bit di stato oltre a un po' di logica per l'interrupt. Ciò non è ancora sufficiente. Otto linee non sono sufficienti per la maggior parte delle applicazioni di input-output. Le applicazioni tipiche richiedono 16 o 24 linee di I/O come minimo. Sarebbe complicato e antieconomico collegare una singola interfaccia da 8 bit per ciascun collegamento di 8 bit. Il chip di interfaccia di impiego generale deve perciò fornire *diversi canali.* Un canale, detto anche *porta*, è un collegamento di 8 bit che può essere usato sia come input o come output. Il PIO deve quindi *multiplexare* un singolo collegamento al data bus del microprocessore su due o tre porte esterne. Ciascuna porta sarà fornita del proprio registro (i) di buffer e delle proprie informazioni di stato. Sarebbe desiderabile fornire il massimo numero possibile di porte di I/O. La limitazione pratica è costituita dal numero di pin (40 pin al massimo ancora una volta). Per questa ragione può venir fornito un massimo di tre porte di input-output. La maggior parte dei dispositivi forniscono due o tre porte di I/O parallelo.

A questo punto abbiamo quasi costruito un dispositivo PIO «standard». La differenza fondamentale tra un PIO e le usuali interfacce standard che venivano in-

serite nei progetti di schede con microprocessore, sta nel fatto che il PIO è programmabile.

Un PIO è programmabile almeno in due modi.

1. Le funzioni della logica di controllo sono programmabili per ciascuna porta. L'utilizzatore può specificare per esempio quale linea verrà usata per il procedimento di handshaking, la direzione nella quale i dati verranno usati e, spesso, la loro funzione. Potrà essere specificato se il segnale di un dispositivo dovrà triggherare degli interrupt oppure no, se dovrà generare dei livelli di soglia o degli impulsi, utilizzare logica positiva o negativa ecc.

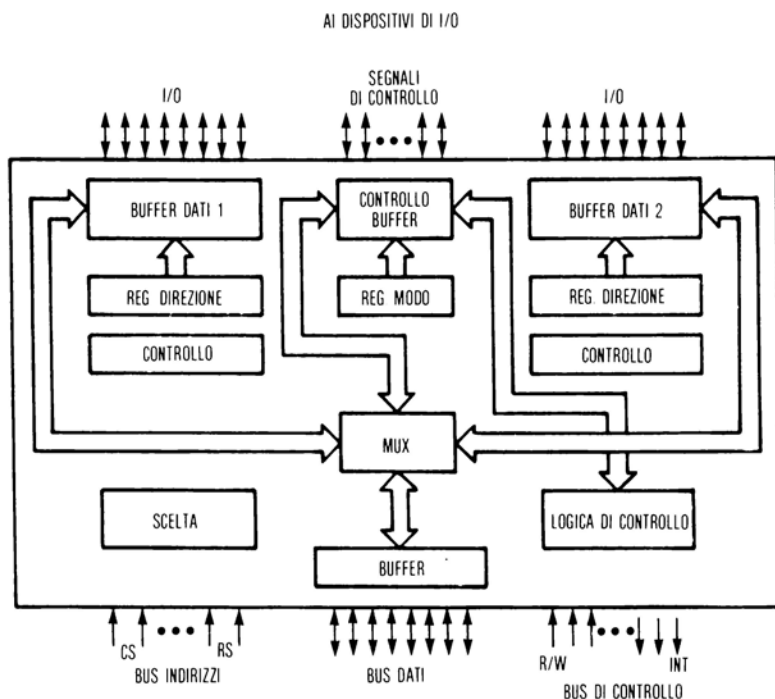



Fig. 3-22: PIO tipico: uno schema semplificato.

2. Un PIO ha delle linee di dati «programmabili». *Ciascuna linea di dato* (o gruppo di linee di dati) su una porta *è programmabile rispetto alla direzione*. Ciascuna linea può essere definita individualmente come linea di input o di output. Questa è un'eresia dal punto di vista di un progettista hardware tradizionale. Utilizzando un PIO è possibile programmare tutte le linee come output alla mattina, cambiare idea e usare tutte le linee come input al pomeriggio. Questa caratteristica

rende il PIO un'interfaccia veramente di impiego generale che può essere usato in qualsiasi situazione standard. È possibile collegare qualsiasi combinazione di linee di input o di output al medesimo PIO o gruppo di PIO. Ora è possibile costruire una scheda microcomputer standard, munita di PIO standard, che possono fondamentalmente interfacciarsi con qualsiasi situazione standard.

La struttura di un PIO compare in Fig. 3-23. Questo PIO ha due porte (8 bit ciascuna) più le linee di controllo. Sono possibili altre combinazioni a scelta del costruttore. Ciascuna porta qui è equipaggiata con tre registri:

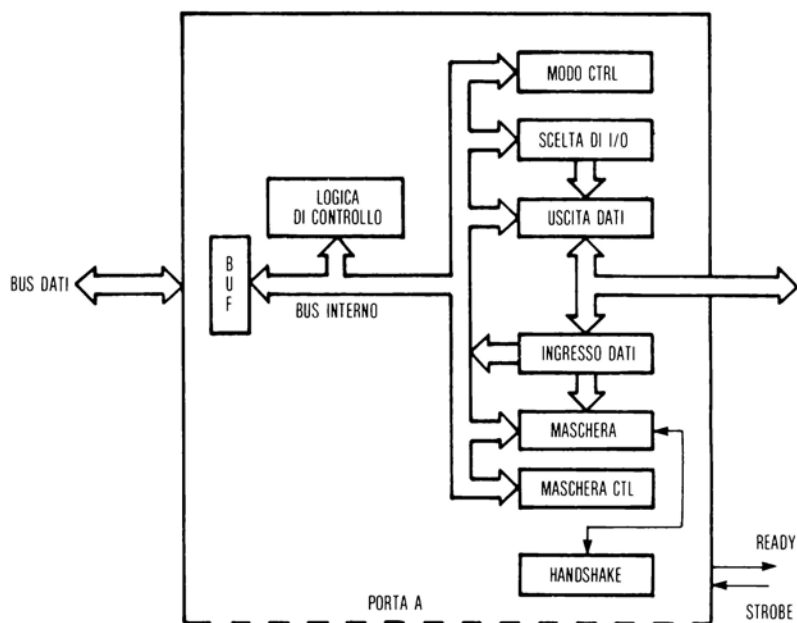



Fig. 3-23: Struttura interna del PIO.

1. Il *registro buffer dei dati* che accumula i dati per l'input o l'output su ciascuna delle otto linee di I/O.

2. *Il registro di direzione*. Uno «0» o un »1» scritto in una posizione di bit di questo registro configurerà la linea corrispondente come input o come output. Lo 0 viene usato generalmente per denotare un input o l'1 un output. Considerando la ortografia di «Input» e «Output» sarebbe stato probabilmente più logico scegliere «1» per Input e 0 per Output. La ragione di questa scelta risiede in un criterio di sicurezza. Quando il sistema viene inizializzato, i contenuti di tutti i registri saranno normalmente resettati a 0. Proprio durante l'accensione del sistema dei segnali spu-

ri potrebbero essere generati e causare danni al mondo esterno. Perciò è di fondamentale importanza che se un qualunque segnale si presenta sulle linee di I/O queste linee risultino configurate come *input* durante lo «start up» (avviamento), piuttosto che come output.

3. *Il registro di controllo.* Questo registro immagazzina i bit di comando emessi dal microprocessore per la porta. Per ciascuna porta il microprocessore specificherà se gli interrupt devono essere generati o meno e quali segnali di controllo verranno logicamente associati alla porta. Ciò significa, in particolare, che ogniqualvolta il buffer dei dati è pieno oppure vuoto verrà settato o resettato entro questo registro di controllo un appropriato bit di stato. Tipicamente questo registro di controllo fornirà la memorizzazione sia della parola di comando emessa dal microprocessore sia della informazione di stato generata dal dispositivo, poichè lo Stato richiede generalmente solo uno o due bit di informazione. Anche la Porta 2 è munita delle medesime possibilità.

Come viene usato un PIO? Per utilizzare il PIO il microprocessore deve eseguire due operazioni di base:

1. Inizialmente caricherà il registro di controllo per specificare il modo in cui i segnali di controllo verranno generati o opereranno.

2. Il microprocessore caricherà il registro di direzione per specificare la direzione nella quale le linee delle porte verranno usate.

Questo deve essere fatto per ciascuna porta. Una volta che i contenuti del registro di controllo e del registro di direzione sono validi il dispositivo può essere utilizzato. Poi verrà operato un trasferimento dei dati mediante un'istruzione standard di trasferimento, come ad esempio l'istruzione MOV che è stata descritta nel caso dell'8080 (nel caso dell'8080 vi sono disponibili, in più, istruzioni specifiche di input e di output). Fino ad ora ci rimangono da risolvere due problemi: Come vengono trasmessi questi dati in registri interni del PIO? Come vengono selezionati i registri?

La risposta dovrebbe risultare ovvia per il lettore attento. Tutti i dati trasmessi dal microprocessore compaiono normalmente sul bus dei dati. I dati che devono essere caricati nei diversi registri del PIO verranno perciò posti sul data bus e simultaneamente verrà eseguita un'operazione di register select (RS) o selezione del registro. La selezione del registro viene realizzata formando un indirizzo sul bus degli indirizzi. Per selezionare il chip (chip select) deve essere fornito almeno un bit e, assumendo che il nostro PIO abbia 8 registri interni o meno, devono essere usate tre linee dell'address bus per fare una selezione 1 di 8 registri (tre linee generano otto configurazioni binarie, cioè permettono la selezione di otto registri possibili).

Riassumendo, il microprocessore selezionerà uno dei registri interni del PIO inviando la configurazione opportuna sul bus degli indirizzi e fornirà gli otto bit del dato da trasferire entro uno di questi registri sul bus dei dati. Un multiplexer inter-

no entro il PIO instraderà i dati di 8 bit al registro appropriato. Il nostro PIO è ora pronto per essere usato. I dati possono ora essere liberamente scambiati tra i suoi buffer dei dati e il microprocessore. Per poter leggere una informazione da un buffer dei dati, il microprocessore (o meglio il programmatore) fornirà il segnale appropriato di RS per selezionare il registro buffer dei dati e generare un ordine di lettura sul bus di controllo. Nel caso di un'operazione di scrittura fornirà semplicemente un segnale «W» sul control bus invece del segnale «R». Per poter leggere lo stato dal PIO, verranno letti i contenuti dell'appropriato registro di controllo. Normalmente una volta che il PIO è stato configurato, cioé una volta che i suoi registri di controllo e di direzione sono stati caricati, non è necessario alcun ulteriore cambiamento e il microprocessore può comunicare direttamente con i buffer dei dati usando una singola istruzione.

Si deve notare che, per quanto possa sembrare complicato ad una prima lettura, il PIO è concettualmente un dispositivo molto semplice. Uno dei suoi vantaggi principali è quello di essere programmabile. La programmabilità di questo dispositivo è di fatto, molto rudimentale. Si può prevedere che la potenza dei PIO verrà accresciuta formandole di un maggior numero di funzioni programmate. Verrà dimostrato più avanti che l'intero concetto di evoluzione dell'LSI è verso l'*integrazione verticale* cioè nella direzione di rendere disponibili più funzioni per chip. Si può facilmente prevedere che il PIO del futuro sarà fornito di processori elementari che renderà disponibili capacità di elaborazione locale.

Esempio: Motorola 6820 PIA

PIA significa «peripheral interface adapter» o adattatore di interfaccia periferico che è il nome usato dalla Motorola per un dispositivo PIO. La struttura del 6820 compare in Fig. 3-24. È essenzialmente identico alla struttura del nostro PIO «standard» di Fig. 3-22. Il PIA è equipaggiato con due porte di input-output (otto bit ciascuna). Ciascuna linea può essere programmata individualmente come input o come output. Uno 0 nel registro di direzione dati specifica un input, un 1 specifica un output. Il 6820 è fornito di due linee di controllo per ciascuna porta di I/O: CA1, CA2 per la porta A, e CB1, CB2 per la porta B. La seconda linea di controllo su ciascuna porta può essere programmata sia come input, sia come output. Per esempio, una delle linee può essere usata come segnale di *Ready* (pronto) e il secondo segnale di *Acknowledge* (riconoscimento). Inoltre il segnale Ready può generare automaticamente un interrupt. Questo componente è provvisto anche di due richieste di Interrupt, IRQA e IRQB, uno per porta.

DOMANDA: *Perchè il Motorola 6820 ha due linee separate di interrupt, considerando che il microprocessore 6800 ha solamente un singolo input di Interrupt?*

La risposta è che, in un sistema semplice, entrambe le richieste di interrupt A e B verranno normalmente fatte confluire sulla singola linea di interrupt. Tuttavia, in un sistema più complesso, per gli interrupt verrà usata una struttura di priorità più elaborata. Tali strutture di priorità verranno descritte più avanti in questo capitolo.

In quel caso ciascuna delle linee di richiesta di interrupt del PIA sarà collegata ad un input di Interrupt separato di un chip di priorità, ad esempio un PIC.

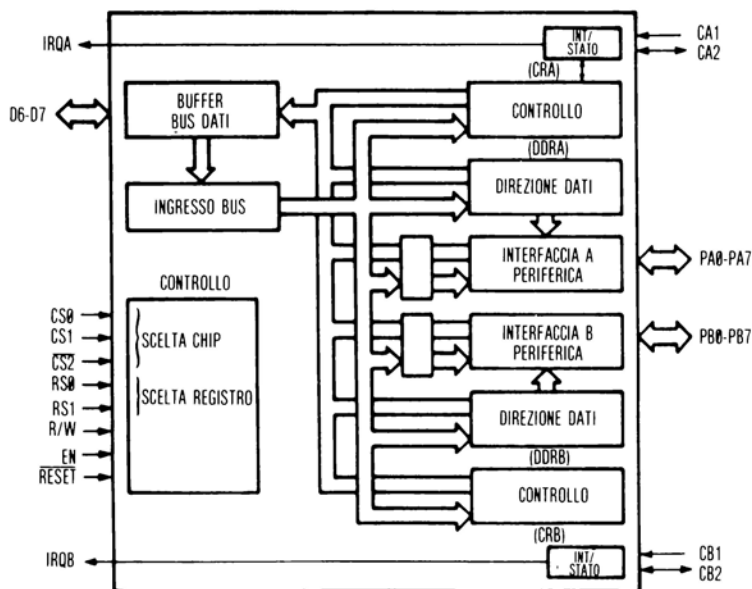



Fig. 3-24: PIA 6820 della Motorola.

Il ruolo dei pin sul 6820 dovrebbe essere chiaro. Due gruppi di segnali sulla sinistra (CS0, CS1, CS2) realizzano una filosofia particolare Motorola. La maggior parte dei componenti della famiglia 6800 hanno tre o più pin di chip select. Verrà dimostrato che in qualsiasi sistema piccolo ciò risparmia l'uso di decodifiche di indirizzo. Tuttavia ciò richiede un maggior numero di pin sul chip. In particolare una delle «caratteristiche» dello schema di selezione dei registri del 6820 è dovuto probabilmente alla mancanza di un numero sufficiente di pin sul dispositivo. Questo corrisponde al secondo gruppo di segnali, RS0 o RS1. Il 6820 è munito di sei registri interni. Sono disponibili solo *due* segnali RS per selezionare uno dei sei registri. Come è possibile questo? Chiaramente due segnali possono selezionare solamente uno di quattro registri o, meglio, uno di quattro indirizzi. Ciò è proprio quello che avviene. Il «registro di interfaccia periferico» e il «registro direzione dati» hanno in comune il medesimo indirizzo su una porta. La selezione tra i due registri è ottenuta mediante un bit entro il registro di controllo. Uno «0» nella posizione di bit 2 del registro di controllo seleziona il DDR (data direction register) e un «1» seleziona il registro di interfaccia periferico. Questa prerogativa viene presentata dal costruttore

come vantaggio: ogni volta che il sistema è inizializzato, cioè quando viene applicato un reset, i contenuti del registro di controllo saranno «0», e sarà selezionato automaticamente il registro di direzione dati. Nessun segnale spurio dovrebbe quindi venir generato sul registro di interfaccia periferico che non è stato selezionato. Ciò è corretto. Tuttavia si potrebbe anche arguire che ciò conduce ad una programmazione piuttosto esagerata. Se si deve accedere in successione al DDR e al registro di interfaccia periferico ciò implica un cambiamento dei contenuti del registro di controllo tra i due accessi.

Il 6820 è un dispositivo di I/O e flessibile e i possibili svantaggi del suo schema di accesso ai registri non dovrebbe essere considerato come un argomento significativo.

ESERCIZIO: *Quali contenuti devono venir caricati in DDRA e in DDRB per configurare la porta A come input e la porta B come output?*

RISPOSTA: *«00000000» specificherà 8 input e «11111111» specificherà 8 output.*

Infine descriviamo un semplice esempio di un processo di handshaking cioè di scambio (letteralmente stretta di mano) tra il 6820 e un dispositivo come un lettore di banda (paper tape reader). Si assume che il paper tape reader sia munito di un buffer a 8 bit. Il microprocessore vuole leggere una parola di dato dal PTR. Esso dovrà dapprima verificare se i contenuti del buffer del PTR sono validi. CA1 verrà

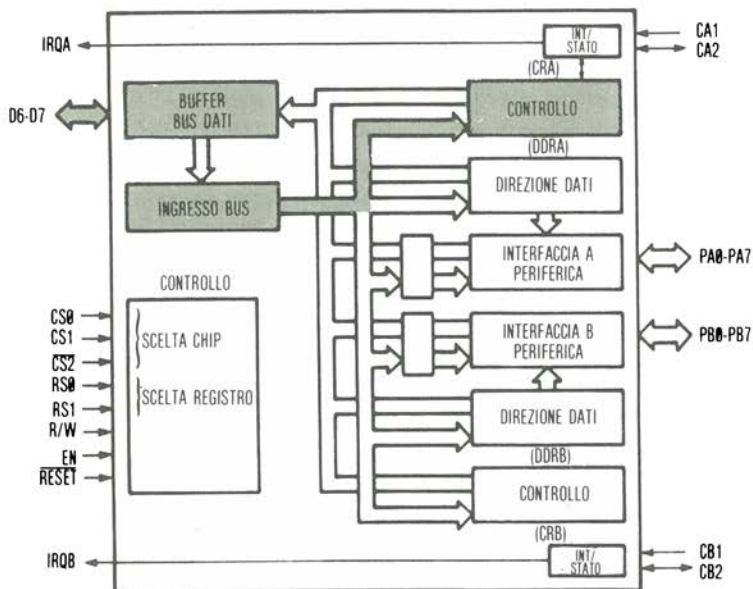



Fig. 3-25: Impiego di un PIA: caricamento del registro di controllo.

usato, per esempio, per collegare il «buffer carico» (o segnale «ready») dal PTR al PIA. Ogni volta che il segnale «ready» verrà ricevuto sarà noto che i contenuti del buffer PTR sono corretti e possono essere letti. I dati presenti sulla porta A possono allora essere letti. Quando ciò avviene un segnale di «acknowledge» o di riconoscimento verrà trasmesso indietro al dispositivo. Questo riconoscimento comunica al dispositivo: «il tuo buffer è stato svuotato, puoi ricaricarti ancora». La maggior parte dei dispositivi è fornita in più di un segnale o flag di «data overrun», che indica che i dati sono stati scritti in un buffer quando i contenuti precedenti non erano ancora stati letti. Questo è un errore che è normalmente verificato in hardware.

Automaticamente, ogni volta che i contenuti del data register sulle porte A o B vengono letti sul data bus del microprocessore, può venir trasmesso un segnale di riconoscimento su CA2 o su CB2 verso la periferia per comunicare a quest'ultima che l'operazione di lettura è stata effettuata.

Similmente un PIA che comunichi con un dispositivo di output, come ad esempio una teletype munita di buffer, deve interrogarlo prima di inviargli qualsiasi informazione. Bisogna verificare che il buffer del dispositivo sia vuoto prima di caricarlo con dati addizionali. Questa volta un segnale di «ready» dal dispositivo acquista il significato: «il mio buffer di ingresso è vuoto, puoi fornirmi il carattere successivo. Vai avanti». Allora il microprocessore può inviare il byte successivo alla periferica.

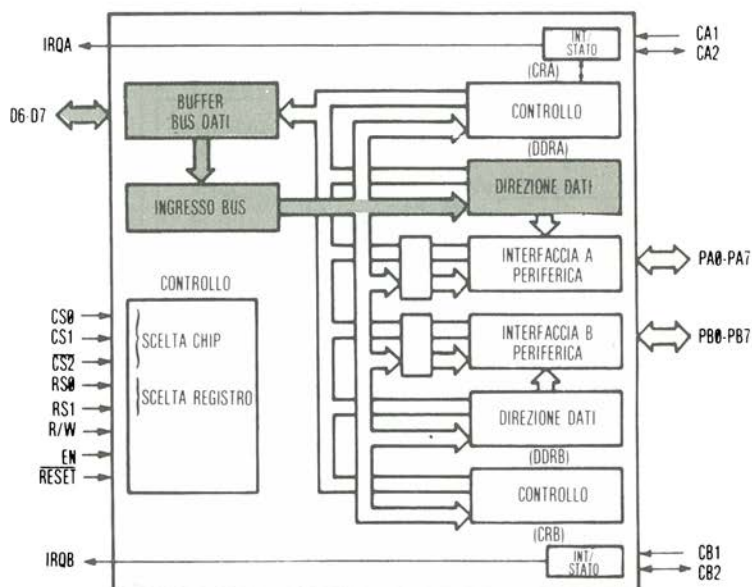



Fig. 3-26: Impiego di un PIA: caricamento direzione dati.

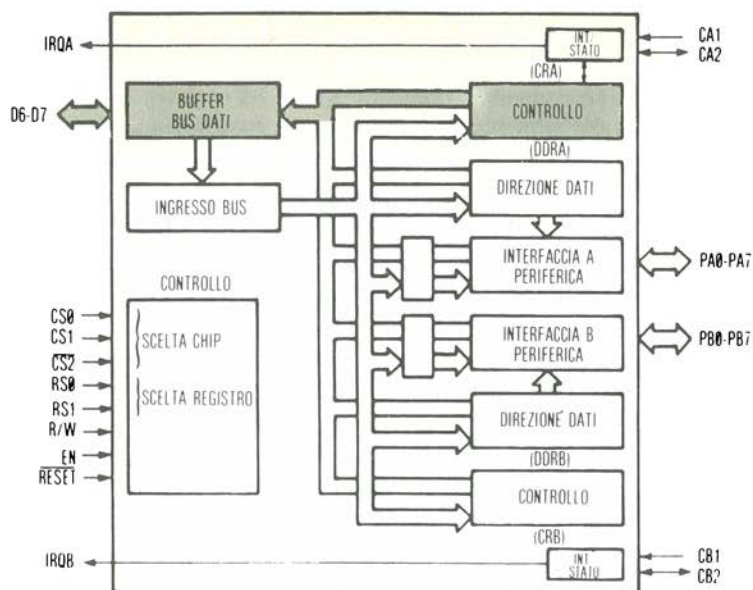



Fig. 3-27: Impiego di un PIA: lettura dello stato.

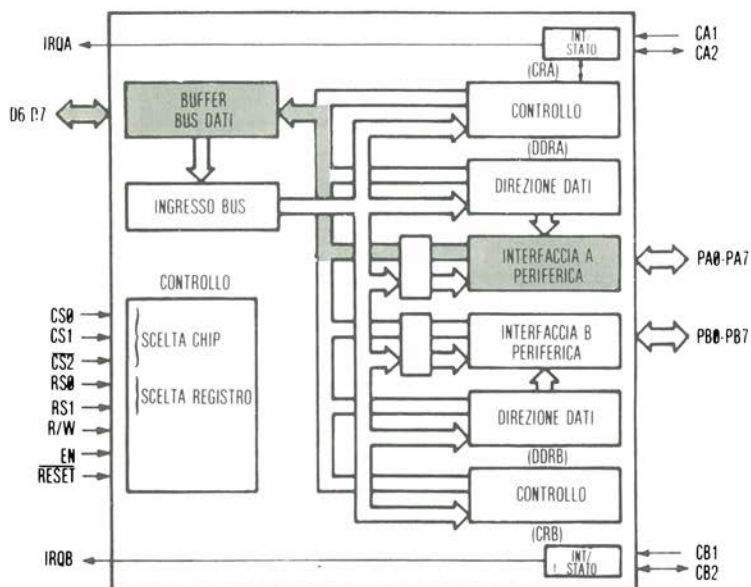



Fig. 3-28: Impiego di un PIA: lettura dell'ingresso.

Esempio: la Intel 8255 PPI

PPI significa «programmable-peripheral-interface» o interfaccia programmabile periferica. Questo dispositivo è fornito di 24 linee di input-output. Queste linee sono normalmente raggruppate su tre porte. A prima vista sembrerebbe che questo dispositivo fosse più potente del PIA che è stato appena presentato. Questo non è proprio esatto. Sebbene questo componente sia munito di tre porte, almeno quattro linee su una delle porte deve essere usata per funzioni di controllo. Da questo punto di vista è fondamentalmente analogo al dispositivo precedente. In più, in questo caso, la PPI non è programmabile per linea, ma per gruppi di quattro linee. Questa non è una mancanza significativa poichè, nella maggior parte dei casi pratici, le linee sono di fatto raggruppate a gruppi di quattro o più. La PPI può essere pro-

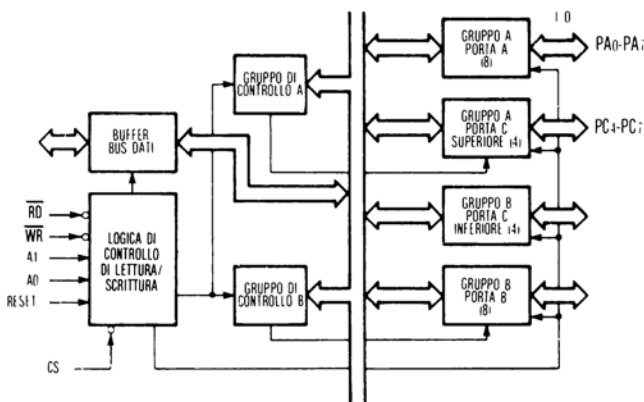



Fig. 3-29: PPI 8255 della Intel.

grammata in tre modi:

1. Modo 0 permette a ciascun gruppo di quattro linee di essere un input o output.
2. Modo 1 programma otto linee come input o come output, entro un gruppo di 12 linee. Le quattro linee rimanenti sono poi riservate per funzioni di controllo.
3. Modo 2 è chiamato modo a «buffer bidirezionale»: sono disponibili otto linee per i dati e cinque linee per l'handshaking.

L'indirizzamento interno dell'8225 viene eseguito nel solito modo. I segnali compaiono in Fig. 3-30. CS è il segnale di chip select, che indica al dispositivo che è stato selezionato. A0 - A1 sono usati in combinazione con le linee «read» e «write» per trasferire le informazioni verso o dai registri della PPI.

$\overline{CS}$	A1	A0	$\overline{RD}$	$\overline{WR}$	FUNZIONAMENTO	
0	0	0	0	1	PORTA A AL BUS DATI	MPU
0	0	1	0	1	PORTA B AL BUS DATI	READ
0	1	0	0	1	PORTA C AL BUS DATI	(A. B. C)
0	0	0	1	0	BUS DATI ALLA PORTA A	MPU
0	0	1	1	0	BUS DATI ALLA PORTA B	
0	1	0	1	0	BUS DATI ALLA PORTA C	WRITE
0	1	1	1	0	BUS DATI AL CONTROLLO	
0	1	1	0	1	NON CONSENTITO	
1					BUS DATI NELLO STATO TRI-STATE	(DISABLE)

Fig. 3-30: Indirizzamento di un 8255.

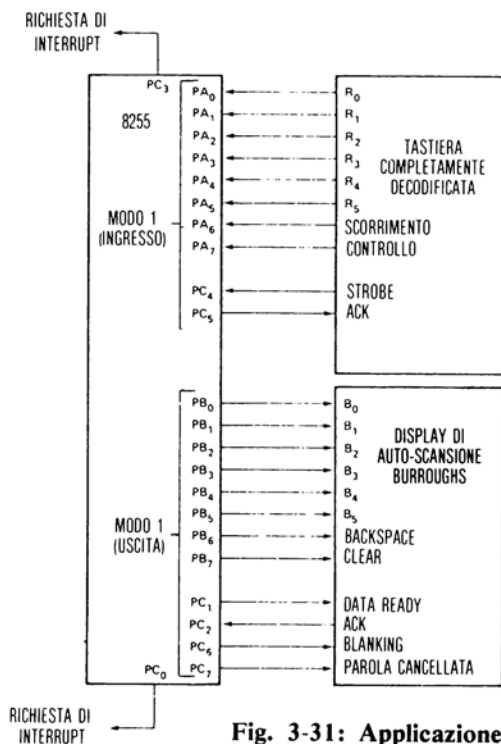



Fig. 3-31: Applicazione tipica dell'8255.

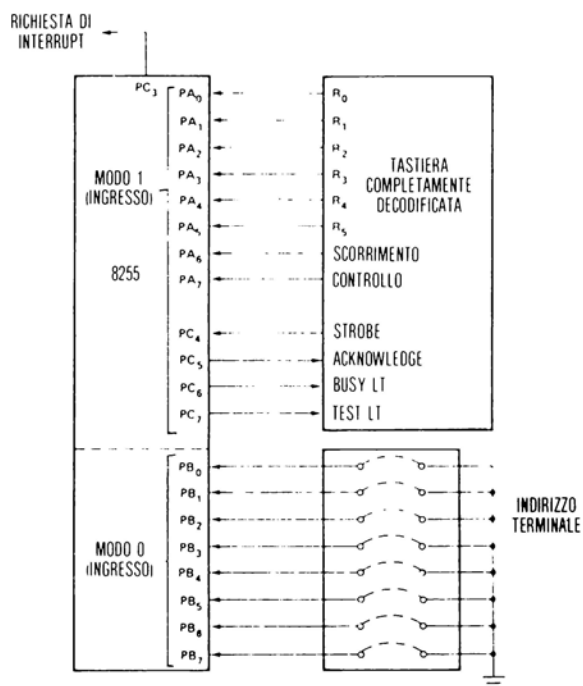



Fig. 3-32: Altra applicazione tipica dell'8255.

Due applicazioni tipiche dell'8255 sono raffigurate in Fig. 3-31 e Fig. 3-32. In figura 3-31, l'8255 è programmato nel modo 1 ed è usato come input. Le quattro linee rimanenti sono la porta C più la porta B, programmate in modo 0. La metà superiore del dispositivo è usata per comunicare con un lettore di banda a 8 bit: la metà inferiore del dispositivo è usata per controllare il funzionamento della macchina utensile.

Esempio: Rockwell PDC e GPI/O

Ogni costruttore di microprocessori che abbia sviluppato una famiglia di componenti attorno ad un microprocessore, oggi ha un PIO o il suo equivalente. La Rockwell ha due dispositivi di questo tipo che sono riportati nelle seguenti due illustrazioni. Si può notare che la Rockwell utilizza anche uno schema di selezione multipla per risparmiare decodifiche esterne di indirizzo.

Altri Chip di Interfaccia I/O di Base

Ora esistono diversi altri componenti combinazione di un PIO, un UART, un microprocessore e persino altri componenti (come un temporizzatore di intervalli o un power fail restart che permette di riinizializzare in caso di interruzioni per man-

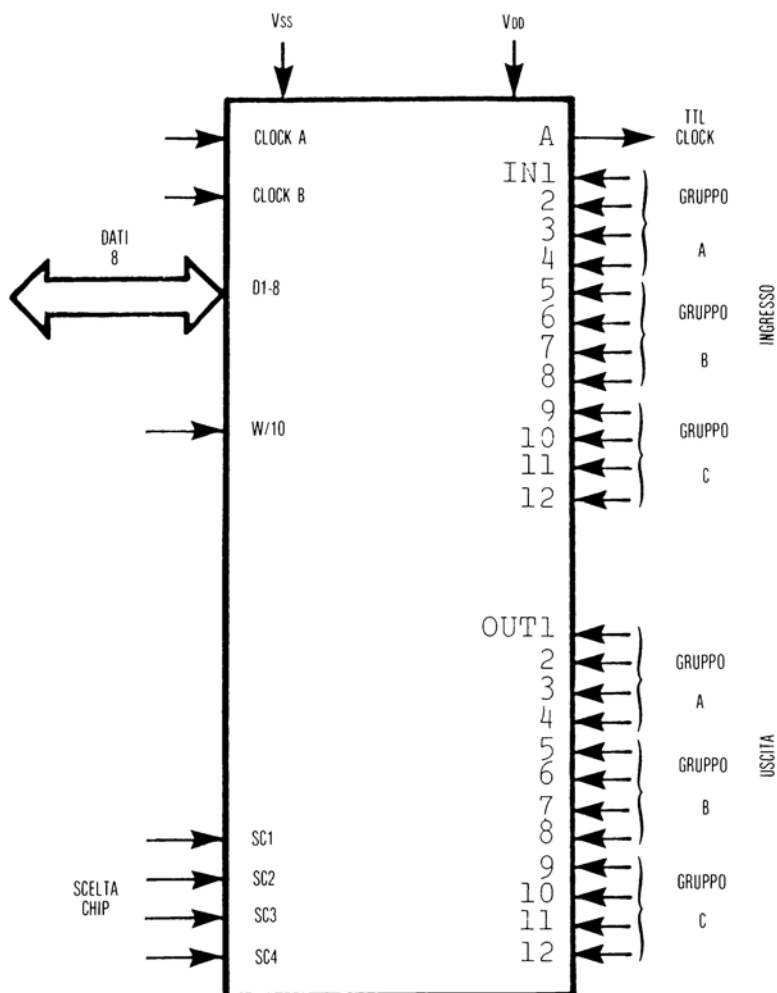



Fig. 3-33: GP I/O della Rockwell.

canza d'alimentazione). Per esempio la Texas Instruments ha introdotto il TMS5501. Il 5501 mette a disposizione metà del PIO con una porta di input a 8 bit, una porta di output a 8 bit più una singola linea seriale asincrona, due interrupt e cinque timer a intervallo programmabile. Questo si interfaccia direttamente all'8080 (non richiede nemmeno l'8228). Questo componente è essenzialmente una miscela di un PIO parziale, un UART parziale, e un PIT (timer a intervallo programmabile).

L'8741, recentemente introdotto dall'Intel, è similmente una miscela di un processore e di funzioni standard di interfaccia.

CHIP DI GESTIONE I/O

Sono state introdotte due classi di chip per facilitare lo scheduling o coordinazione dei processi di input o output. Essi sono, rispettivamente, il PIC (controllore di priorità di interrupt) e il DMAC (controllore dell'accesso diretto in memoria).

Gestione dell'Interrupt

È stato mostrato all'inizio di questo capitolo come gli interrupt possano essere usati per ottenere tempi di responso rapidi da un microprocessore allo scopo di servire meglio dei dispositivi di I/O. Sono stati incontrati essenzialmente due problemi:

1. Possono venir richiesti interrupt simultanei. Questo problema verrà risolto mediante uno schema di *priorità* che verrà descritto in seguito.

2. La disponibilità di una singola linea di interrupt impone la necessità di identificare il dispositivo che ha innescato l'interrupt. Questo è il *problema dell'identificazione dell'interrupt* che risolveremo ora.

In considerazione della limitazione sul numero di pin per package si è stabilita che la maggior parte dei microprocessori abbia uno o due (eventualmente di più) linee di interrupt. Ciò non è sufficiente per dedicare una linea di interrupt ad un dispositivo. Conseguentemente diversi dispositivi verranno collegati a questa medesima linea di interrupt. Non appena viene inviato un segnale di interrupt al microprocessore, sarà necessario determinare quale dispositivo ha causato l'interrupt, per permettere al microprocessore di eseguire la routine corretta di trattamento dell'interrupt. Si possono usare due metodi fondamentali:

1. *Il metodo software* avrà un programma di polling che interrogherà a turno ciascuno dei dispositivi collegati alla linea di interrupt per determinare quale sia il dispositivo che richiede l'interrupt. Nella sua forma più semplice la routine di interrupt leggerà un bit di stato di interrupt su ciascun dispositivo per determinare se questo è l'origine dell'interrupt. Se il dispositivo ne ha fatto richiesta la routine eseguirà allora un branch all'appropriato esecutore di interrupt. L'identificazione del dispositivo che causò l'interrupt può essere assistito da hardware esterno. Il metodo *daisy-chain* o a catena di margherita, utilizza una linea in più dal microprocessore verso il dispositivo di I/O, poi dal dispositivo I/O 1 a quello di I/O 2, poi dal dispositivo di I/O 2 all'I/O 3 e così via fino a ritornare al microprocessore. Su questa linea viene generato un segnale. Il dispositivo 1 riceve il segnale. Se non aveva generato un interrupt esso lascia che il segnale fluisca verso il dispositivo di I/O successivo. Qualora il segnale raggiunga un dispositivo che aveva innescato un interrupt, il dispositivo porrà un numero di identificazione («vettore») sul data bus dove può essere letto dal microprocessore. Per aumentare l'efficienza di questo schema si possono usare altri metodi ancora.

2. *Il metodo hardware* è molto più efficiente. Lo schema software è semplice in

termini di collegamenti richiesti, ma richiede un tempo notevole dal microprocessore aumentando la quantità di operazioni da svolgere per rispondere ad un interrupt. Lo schema hardware attribuisce la responsabilità di fornire automaticamente l'indirizzo dell'appropriata routine di gestione dell'interrupt ad un dispositivo hardware. Questa è chiamata la tecnica dell'*interrupt vettorizzato* automatizzato. Un interrupt vettorizzato è un interrupt che fornisce l'indirizzo di branching alla routine di gestione dell'interrupt nello stesso tempo. Per fornire questa possibilità il componente che gestirà l'interrupt dovrà essere fornito di un registro di indirizzamento a 16 bit per ciascun dispositivo. Questo registro di indirizzamento conterrà l'indirizzo di branching richiesto. Il funzionamento dettagliato verrà presentato nel paragrafo seguente.

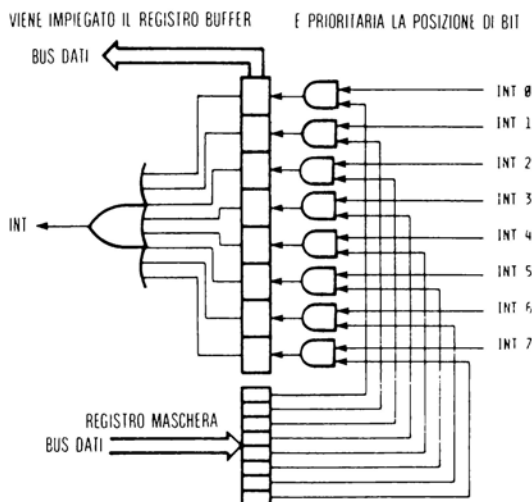



Fig. 3-34: Circuito di base della direzione priorità.

Priorità

Il problema rimanente è di determinare quale dispositivo di I/O dovrebbe essere abilitato nel caso avvenissero richieste di interrupt simultanee. La soluzione è di realizzare uno schema di priorità. Nella sua forma più semplice, ma più frequentemente utilizzata, lo schema di priorità assegna un numero fisso di priorità a ciascun dispositivo. Per esempio il floppy disk avrà livello di priorità 1, una segnalazione di guasto all'alimentazione avrà livello di priorità 0. Si deve notare che la priorità 0 è normalmente la più alta nel sistema. Le priorità non necessariamente devono venir fissate e alcune PIC più recenti permettono all'utilizzatore di modificare le priorità mediante programma. Tuttavia la gestione di priorità, che possono variare dinamicamente, implica una struttura software talmente complessa, che viene usata raramente.

Un miglioramento allo schema a priorità fissata è diventato uno standard nella gestione degli interrupt: questo miglioramento è la possibilità di *mascheratura*. Ogni dispositivo per il trattamento hardware degli interrupt mette a disposizione un registro di mascheratura. Questo si potrà spiegare meglio su un circuito reale.

Il circuito di base per il trattamento degli interrupt con l'esclusione del codificatore di priorità e della possibilità di vettorizzazione, compare in Fig. 3-34. Questo circuito gestirà otto interrupt denominati INT0-INT7, che compaiono sulla destra della figura. Il registro di mascheratura compare in fondo alla figura. Ogni volta che i contenuti di un bit della maschera sono 0, come si può vedere, questo «0» bloccherà la propagazione del segnale di interrupt verso la parte sinistra della figura. Il livello di interrupt corrispondente viene allora detto mascherato. La presenza di un bit a «1» nel registro di mascheratura permetterà all'interrupt corrispondente di propagarsi verso sinistra. Se devono essere usate tutte le linee di interrupt, oppure abilitate, il registro di mascheratura conterrà tutti «1». Se la linea di interrupt 2 deve essere ignorata, il bit 2 del registro di mascheratura verrà posto a «0». I livelli di interrupt che non sono stati mascherati potranno settare un bit in un registro di interrupt sulla sinistra della figura. I contenuti di questo registro possono essere letti al di fuori del circuito sul data bus (in cima alla figura). Verrà mostrato più avanti come questo registro permetta una facile realizzazione di una decodifica di priorità software. Infine un OR inclusivo delle linee di questo registro fornisce il segnale di interrupt finale sulla sinistra di questa figura. Questa richiesta di interrupt è collegata alla linea di interrupt del microprocessore.

Nel funzionamento normale il registro di mascheratura è caricato dal programmatore con l'adatta configurazione di bit che abiliterà i livelli di interrupt selezionati. Assumendo che vengano usate tutte le linee di interrupt, la maschera conterrà tutti 1. Se uno o più interrupt vengono innescati essi si propagheranno verso la sinistra della figura e ne risulterà una «richiesta di interrupt». Il microprocessore leggerà allora i contenuti del registro di interrupt e troverà un «1» in ciascuna posizione di bit dove un dispositivo ha richiesto il servizio. Assumendo che diversi disposi-

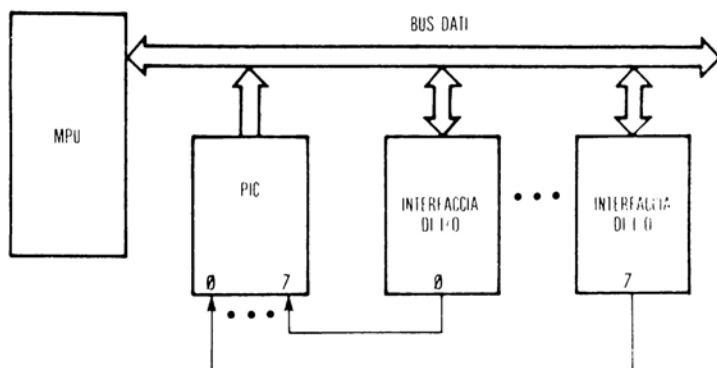



Fig. 3-35: Il PIC intercetta e dirige gli interrupts.

tivi abbiano richiesto servizio simultaneamente è facile allora realizzare uno schema di priorità. Assumeremo qui che l'interrupt 0 sia quello a più alta priorità, l'interrupt 1 il successivo e così via. Il microprocessore testerà allora semplicemente il bit 0 del registro di interrupt, poi il bit 1, poi il bit 2 fino a che trova un «1». Non appena trova un «1» il livello di interrupt corrispondente verrà servito. Questo garantisce che il livello più alto di interrupt venga servito per primo. Dopo che questo interrupt è stato servito il microprocessore dovrà leggere nuovamente i contenuti del registro di interrupt, se il segnale INT è ancora attivo, al fine di servire gli altri interrupt in attesa. Questa è una semplice realizzazione software di uno schema di interrupt a priorità.

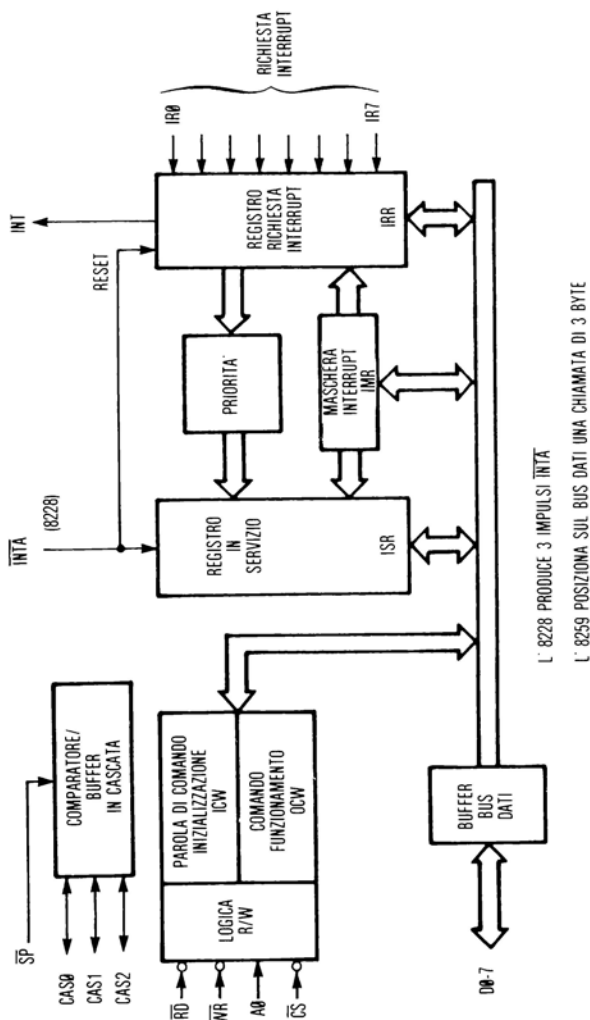



Fig. 3-36: Architettura del PIC 8259.

Siamo pronti ora per il funzionamento reale di un chip completo di un controllore di priorità di interrupt.

Esempio: Intel 8259 PIC

L'architettura dell'8259 compare in Fig. 3-36. Esso fornisce una completa gestione degli interrupt compreso il trattamento delle priorità, la mascheratura degli interrupt e la vettorizzazione automatica degli interrupt. Esso è realizzato su un dispositivo statico NMOS a 28 piedini. In più il dispositivo può essere messo in cascata: esso può essere associato con un massimo di altri otto PIC per gestire 64 livelli di interrupt separati. Esaminiamo qui in maggior dettaglio il trattamento degli interrupt dell'8080.

L'8080 da solo fornisce possibilità di interrupt minime. In risposta ad un segnale di interrupt, purchè gli interrupt siano abilitati, l'8080 accetterà l'interrupt e invierà indietro un segnale «interrupt acknowledge» di riconoscimento sul piedino INTA. (Gli interrupt sono abilitati mediante la linea INTE). Il microprocessore non prende ulteriori iniziative e si ferma in uno stato speciale nel quale esso attende che un'istruzione venga forzata sul data bus. Non appena l'istruzione compare sul data bus essa verrà trasmessa direttamente nel registro istruzione dell'unità di controllo, ove essa verrà decodificata. Questo è il trattamento minimale degli interrupt. L'8080 non preserva automaticamente lo stato del program counter. È responsabilità della logica esterna fornire un'istruzione al microprocessore dopo avergli fornito l'INTA. Poiché i PIC in un solo chip sono stati introdotti solo recentemente, la soluzione usuale di questo problema era di forzare un'istruzione speciale sul data bus: la restart (RST). Il vantaggio dell'istruzione RST è quello di essere una speciale chiamata di 1 byte. Una tipica chiamata di programma richiederebbe tre byte. L'istruzione RST è munita di un campo di 3 bit che realizzano una chiamata diretta ad una delle otto locazioni di memoria: 0, 8, 16, ... fino a 256. La routine di trattamento dell'interrupt sarebbe normalmente memorizzata ad uno di questi indirizzi. Inoltre l'istruzione RST preserva automaticamente il program counter sullo stack. Tuttavia è responsabilità della gestione dell'interrupt preservare qualunque degli altri registri interni dell'8080. Questa, fino ad ora, era una deficienza dell'8080 e il responso agli interrupt era lento poichè la maggior parte del trattamento doveva essere eseguito mediante software.

Il nuovo PIC ha migliorato significativamente la situazione. L'8259 fornisce essenzialmente un trattamento dell'interrupt altamente automatizzato, compresa la vettorizzazione automatica. Esso non realizza, tuttavia, nessuna preservazione automatica dei registri dello stack. È ancora responsabilità del programmatore realizzarlo nell'ambito della gestione dell'interrupt. In risposta ad un INTA dall'8080, l'8259 pone un'istruzione di 3 byte sul data bus in modo automatico: essa è una chiamata (11001101) seguita da due byte. I due byte contengono l'indirizzo del programma gestore dell'interrupt (interrupt handler). Essi sono gli *indirizzi vettorizzati*. L'8259 è fornito di otto registri di 16 bit contenenti i vettori di interrupt. Questi registri devono essere caricati dal programmatore prima di essere usati. Sa-

ranno necessarie due operazioni per caricare il registro di 16 bit prima che il livello di interrupt corrispondente possa essere usato. Tuttavia fondamentalmente questo è compiuto durante la fase di inizializzazione del sistema, una volta per tutte. I contenuti normalmente non vengono mutati durante il funzionamento del sistema e continueranno a fornire il medesimo indirizzo di 16 bit per ciascun livello di interrupt. Questo è chiaramente il modo più efficiente di procedere per ottenere la vettorizzazione degli interrupt. Ad esempio se un interrupt deve avvenire a livello 4 e assumendo che il suo livello di priorità sia abilitato, i contenuti del terzo registro di indirizzamento da 16 bit verrà usato per fornire la vettorizzazione automatica.

Inoltre l'8259 include l'usuale mascheratura degli interrupt, la cui funzione è stata descritta: qualsiasi degli otto livelli interrupt può essere inibito ponendo uno «0» nella posizione di bit appropriata dalla maschera.

Esso include anche una possibilità di priorità che permette di inibire automaticamente qualsiasi interrupt avente priorità più bassa di un livello specificato. Il livello di priorità scelto viene caricato nel «registro in servizio». Un semplice comparatore paragona la priorità dell'interrupt al livello memorizzato in questo registro, prima di permettere che questo interrupt si propaghi all'esterno.

Infine sono stati previsti diversi modi di funzionamento del dispositivo che realizzano vari schemi di ordinamento delle priorità.

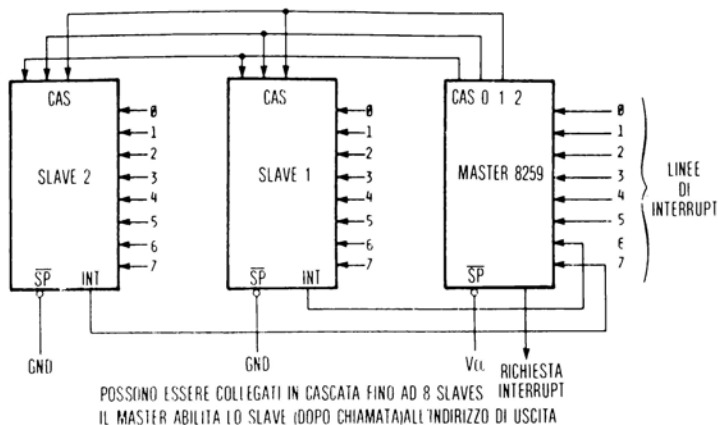



Fig. 3-37: I PIC possono essere collegati in cascata.

Priorità Intel a confronto con Motorola

È interessante paragonare l'approccio scelto nell'8080 con quello realizzato nel 6800 poiché questi due componenti sono direttamente in concorrenza. Il 6800 realizza una filosofia di interrupt completamente differente. Gli interrupt sono abilitati o disabilitati da un bit interno di un registro di maschera che può essere settato dal

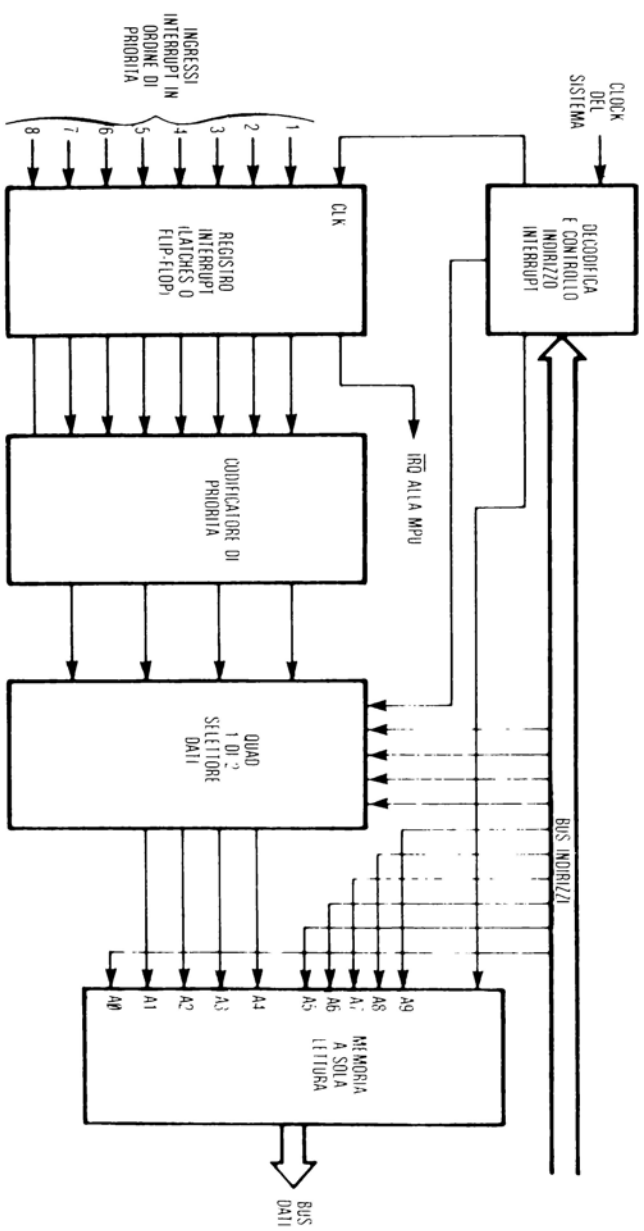



Fig. 3-38: Interrupt a priorità vettorizzata per il 6800.

programmatore (mediante una linea). Ciò ha il vantaggio di risparmiare un pin del microprocessore (usato dall'8080 per la linea INTE). La differenza principale sta nel responso automatico agli interrupt: in risposta ad un interrupt e supposto che sia abilitato, il 6800 reagisce conservando *tutti i suoi registri interni* automaticamente nello stack e realizzando un branching ad una locazione riservata di memoria ed esegue automaticamente un branch all'indirizzo di 16 bit che era contenuto in FFF8, FFF9. Il 6800 preserva automaticamente tutti i registri macchina. In più esso esegue un branch automaticamente ad un indirizzo fisso in memoria. Nei casi normali ciò permette un più veloce responso all'interrupt rispetto all'8080. La complessità interna della logica richiesta entro il microprocessore è naturalmente maggiore. Ma questo approccio ha anche degli svantaggi. Lo schema usato dalla Motorola non si presta facilmente alla vettorizzazione automatica.

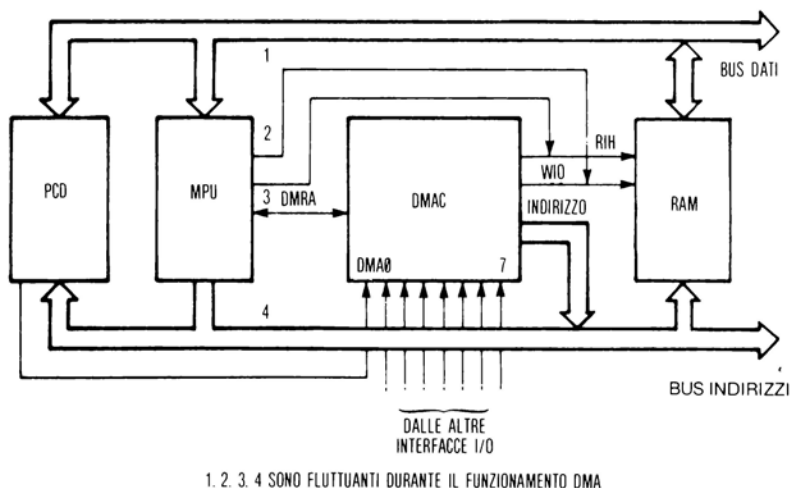



Fig. 3-39: Impiego di un DMAC (Rockwell).

Ogni volta che avviene un interrupt ci sarà un branch automatico all'indirizzo, qualunque esso sia, memorizzato in FFF8, FFF9. Una routine software dovrebbe poi determinare qual'è il gestore finale dell'interrupt che dovrebbe essere attivato. Per ovviare a questo problema è stato perciò introdotto dalla Motorola un nuovo dispositivo, un PIC speciale. Naturalmente esso tratta otto livelli di interrupt come al solito. Per fornire la vettorizzazione automatica esso controlla l'address bus. Ogni volta che il dispositivo riconosce «FFF8, FFF9», prende il controllo del data bus. Invece di permettere che la *memoria* fornisca i contenuti di FFF8-FFF9, è il *PIC* che fornirà il corretto indirizzo di branch. Il PIC è fornito di otto registri di 16 bit e in funzione del livello di interrupt che è stato attivato, essa genererà l'indirizzo di

branching corretto per quel livello. In altre parole questo PIC sostituirà l'indirizzo corretto di 16 bit per uno degli otto livelli di interrupt al posto dell'indirizzo di 16 bit che era contenuto in memoria all'indirizzo FFF8-FFF9. Il problema è così risolto. Questo approccio è chiaramente molto efficiente.

Questo dimostra anche che il PIC Motorola è incompatibile con il PIC della Intel. I benefici risultanti sono essenzialmente simili, fatta eccezione per il fatto che il 6800 preserva automaticamente tutti i registri macchina. Nel caso dell'8080 questo deve essere realizzato mediante una breve routine software se è richiesto. In alcuni casi non è necessario memorizzare tutti i registri macchina. In quel caso l'8080 risulta migliore nel confronto delle prestazioni. Si deve anche precisare che l'8080 ha un maggior numero di registri interni rispetto al 6800. Conservare tutti i registri interni nel caso dell'8080 potrebbe essere indesiderabile e inefficiente.

Il controllore di Accesso Diretto in Memoria (DMAC)

I principi del direct-memory-access sono stati presentati all'inizio di questo capitolo. Diverse tecniche di DMA vengono utilizzate per permettere ad una periferica di comunicare direttamente con la memoria. Il processore può venir bloccato con un comando di HALT dal DMA, o sospeso, oppure il DMA può impadronirsi di un ciclo di memoria del microprocessore oppure ancora possono venir usate combinazioni di questi metodi. Il funzionamento di un DMAC verrà illustrato con un esempio. In Fig. 3-39 compare un sistema utilizzante il DMAC Rockwell. La sequenza di funzionamento è la seguente:

1. Il PDC (PIO della Rockwell collegato alla periferica) richiede servizio dal DMAC sulla linea DMA0 (la linea a più alta priorità).

2. Il DMAC trasferisce la richiesta al microprocessore (MPU) sulla linea DMRA.

3. La MPU termina l'istruzione che stava eseguendo (eccetto gli I/O) e rispedisce indietro un segnale di riconoscimento sulla linea bidirezionale DMRA, verso il DMAC. La MPU è ora entrata in uno stato di WAIT. Essa lascia andare il data bus e l'address bus nello stato di alta impedenza o «flottante» (di qui la necessità di bus tri-state sui microprocessori).

4. Il DMAC invia un segnale di «acknowledge» al PDC informandolo che il processore è stato sospeso e che il trasferimento può avvenire.

5. Il DMAC invia gli appropriati indirizzi di trasferimento sull'address bus. Il DMAC è fornito internamente di otto registri da 16 bit che provvedono l'indirizzo di inizio della parola o del trasferimento di un blocco alla memoria. Il DMAC contiene dei registri addizionali, come un registro contatore (per ciascun livello DMA), che specifica quante parole devono venir trasferite. Si deve notare che i contenuti di questi registri devono essere precaricati dal programma prima di poter essere usati. Il DMAC sostituisce la propria logica al microprocessore e fornisce un indirizzo al-

l'address bus per l'accesso alla memoria.

6. Il DMAC fornisce poi un segnale «read» o «write» su RIH o su WIO. A questo punto la memoria ha ricevuto il proprio indirizzo e il proprio ordine di «read» o di «write». Se è stato specificato un ordine di «write», è responsabilità del PDC fornire la parola di dato alla memoria.

7. Il PDC, cioé il dispositivo di I/O, può ora far entrare o inviare in uscita la propria parola di dato.

8. Dopo ciascun trasferimento di parola il DMAC incrementa automaticamente il suo address register interno e aggiorna il suo contatore di parola.

Il trasferimento del blocco continua fino a che:

- Il dispositivo di I/O lascia cadere la propria richiesta di DMA.

- Il contatore di parole arriva al valore 0. Allora è stata raggiunta la fine del blocco di parole specificato e il trasferimento viene fermato.

- La metà inferiore dell'address register interno del DMAC scatta dal valore «11111111» al valore «00000000». Questa è una caratteristica speciale del dispositivo dovuta all'organizzazione per pagine della sua memoria. Ogni volta che si attraversa un confine di pagina, cioè un multiplo di 256, un registro di pagina deve venir incrementato. Il trasferimento in DMA deve essere interrotto.

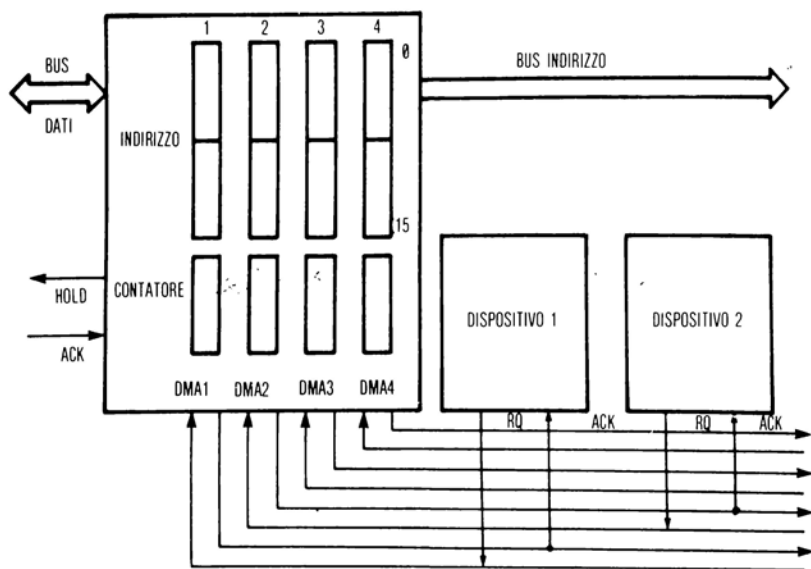



Fig. 3-40: Il DMAC richiede un registro di indirizzo a 16 bit ed un contatore ad 8 bit per ogni canale.

- Avviene una richiesta di interrupt a livello 0 all'MPU. Il livello 0 di interrupt corrisponde normalmente ad una mancanza di alimentazione della maggior parte dei sistemi. In tal caso sono ancora disponibili pochi millisecondi di tempo di elaborazione per preservare il più possibile dello stato del sistema. Normalmente questo tempo viene usato per preservare i contenuti dei registri interni del microprocessore e per interrompere l'I/O in modo ordinato. Il trasferimento DMA, come tutte le altre operazioni di I/O, è immediatamente fermato e si verificherà un branch ad una speciale routine di alimentazione per un'ordinata interruzione. La conservazione dei contenuti dei registri interni del microprocessore naturalmente è ragionevole solo se la memoria è sostenuta da una batteria, cioè se i contenuti della RAM non vengono perduti per la durata dell'assenza di alimentazione.

- Il DMAC riceve una richiesta ad un più alto livello di priorità. Il più alto livello di priorità è DMA0; il più basso DMA7. Ad esempio assumendo che si stesse onorando una richiesta a DMA3, qualsiasi richiesta su DMA0, DMA1 o DMA2 determinerebbe una sospensione di DMA3. Il livello di priorità più alto verrebbe onorato. Non appena fosse completamente trasferito il blocco a più alto livello di priorità, verrebbe ripreso automaticamente il trasferimento al livello inferiore.

Altri tipi di DMAC

Costruttori diversi hanno introdotto le loro versioni personalizzate di DMAC adatte ai loro sistemi. La Intel ha introdotto l'8257 che compare in Fig. 3-41.

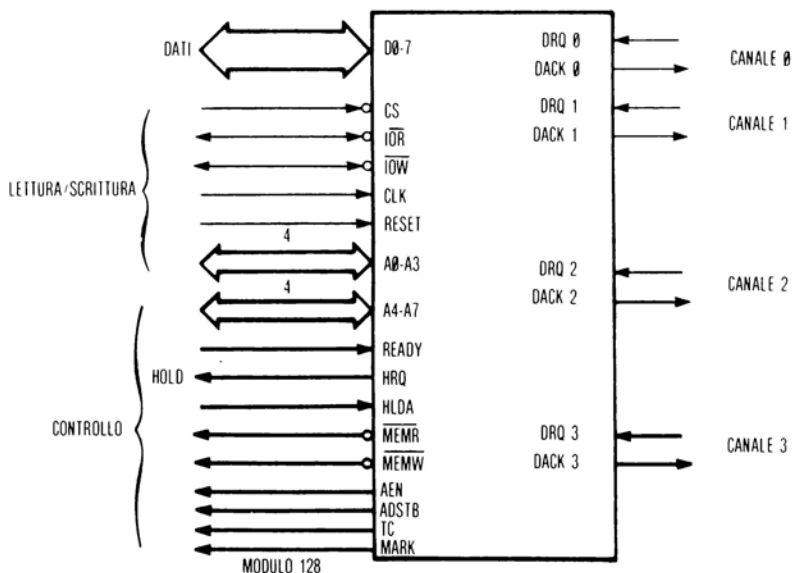



Fig. 3-41: DMA 8257 della Intel.

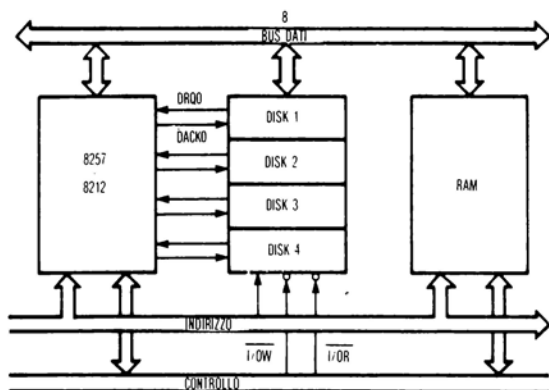



Fig. 3-42: Applicazione tipica dell'8257.

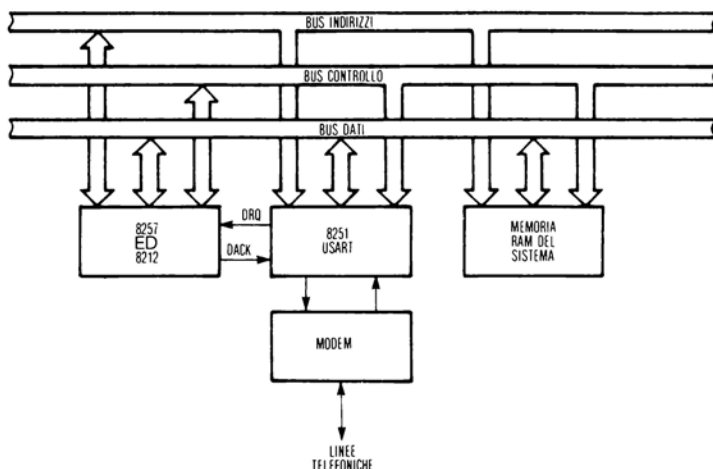



Fig. 3-43: Impiego di un DMA per una comunicazione ad elevata velocità.

L'8257 richiede in aggiunta un latch esterno (8212) per poter preservare otto bit di indirizzo. Un esempio di una possibile applicazione dell'8257 appare in Fig. 3-42. Vengono forniti solo quattro livelli di DMA e l'8257 mostrato in figura controlla quattro diverse unità disco, ciascuna connessa al proprio livello di DMA.

CHIP AUSILIARI

Sono stati sviluppati, o verranno introdotti prossimamente, diversi chip ausiliari che forniscono possibilità in hardware che precedentemente richiedevano dei com-

ponenti addizionali esterni o delle elaborazioni software. Un chip ausiliario utilizzato per sequenzializzazioni è il temporizzatore a intervalli programmabili (PIT). Un PIT è fornito di diversi contatori indipendenti. Esso può funzionare agendo sugli input, sugli output o su entrambi. In output mode esso misura la durata di uno o più impulsi esterni. Sull'output un registro contatore viene precaricato con un valore espresso in millisecondi o in microsecondi e ogni volta che il contatore arriva a 0 viene settato un bit di stato oppure viene generato un interrupt. In altre parole viene generato un segnale al termine di un precisato periodo di tempo.

Il vantaggio fondamentale di un PIT è di migliorare la capacità di elaborazione del microprocessore. Nella maggior parte delle applicazioni si rende necessario che il microprocessore misuri un tempo trascorso sull'input o sull'output mediante tecniche di conteggio. Questo è un compito che occupa molto tempo. La disponibilità di un componente hardware libera il microprocessore rendendolo libero per altri compiti. In più un PIT è indispensabile usualmente nel caso di un funzionamento in tempo reale. Nel caso di un sistema che utilizzi degli interrupt non si possono più usare contatori software che forniscano temporizzazioni con qualche grado di affidamento sulla precisione. Un contatore potrebbe venir interrotto in qualunque momento da un evento esterno che determinerebbe così delle misure errate di tempo. Tuttavia un PIT esterno significa aggiungere un chip in più. La vera soluzione naturalmente sarebbe quella di realizzare un PIT direttamente sul chip del microprocessore. Si può prevedere che ciò verrà fatto per la maggior parte dei chip MPU che verranno introdotti in futuro: il PIT dovrebbe venir realizzato sullo stesso chip del microprocessore stesso. Comunque per rimediare a questo inconveniente spesso viene realizzato un PIT in uno degli altri chip che sono presentati nel sistema come un PIO o un UART.

Ad esempio il PIT 8253 della Intel fornisce tre contatori indipendenti da 16 bit, una velocità da 0 a 2 MHz, conteggio in binario o BCD in sei modi di funzionamento programmabili dall'utilizzatore.

CHIP PERIFERICI DI CONTROLLO

La terza categoria di chip di interfaccia che devono essere descritti in questo capitolo è quella dei controllori diretti di periferiche. Tradizionalmente il controllo di complessi dispositivi meccanici ha richiesto una o più schede di logica collegata direttamente al dispositivo al fine di collegarlo a qualunque tipo di processore. Sino dal 1976 è diventato possibile realizzare chip di controllo completi, ancorché semplificati, per la maggior parte dei dispositivi usuali interfacciati ad un sistema a microprocessore. In particolare ci sono oggi controllori di visualizzatori, controllori di tastiere, controllori di stampanti, controllori per telecomunicazioni, controllori di dischi, controllori per cassette e controllori di CRT.

In generale si può prevedere che saranno disponibili interfacce su 1 chip per ogni tipo significativo di periferica che potrà venir interfacciata con un microprocessore.

Questo è lo stadio finale dell'integrazione completa di tutto il sistema in LSI. Tipicamente un chip di controllore sarà interfacciato direttamente ad un PIO o a diversi PIO o ad un UART, e collegato direttamente entro il sistema. Per uno studio dettagliato del controllo periferico e dei chip controllori si rimanda il lettore interessato al nostro volume sulle Tecniche di Interfacciamento. La vastità dei dettagli connessi a questo argomento andrebbe al di là del livello di questo volume.

DISPOSITIVI TIPICI DI I/O PER IL MICROPROCESSORE

Si possono distinguere tre tipi di dispositivi: dispositivi di input, dispositivi di output e dispositivi per la memoria di massa (o «bulk-memory»).

Il dispositivo di input più frequentemente usato nel mondo dei microprocessori è la tastiera esadecimale. Una tastiera esadecimale è una tastiera a 16 tasti come quelle usate nei calcolatori tascabili. In produzioni di serie tali tastiere costano meno di un dollaro. Probabilmente la tastiera esadecimale diverrà un dispositivo universale presente virtualmente su ogni tipo di apparecchiatura di grande consumo. La maggior parte dei dispositivi di largo consumo ora incorpora dei microprocessori, a partire dai televisori alle macchine lavatrici; essi usano una tastiera come dispositivo di input. Ogni utilizzatore o utilizzatrice di elettrodomestici dovrà perciò ben presto collegarsi al proprio elettrodomestico mediante un simile dispositivo.

I dispositivi di uscita più frequentemente usati sono tre: due di questi sono i visualizzatori a diodi emettitori di luce (LED) e i visualizzatori a cristalli liquidi (LCD). Questi sono visualizzatori (display) a 7 segmenti che permettono la rappresentazione numerica da 0 a 9 nonché delle lettere da A ad F. Essi possono quindi rappresentare ciascuna delle 16 unità esadecimali. Sono i dispositivi di visualizzazione più economici tra quelli disponibili e sono utilizzati estensivamente su orologi digitali, calcolatori tascabili ed altri dispositivi a basso costo. Poiché ciascun LED o LCD può visualizzare tutte le unità esadecimali, un sistema di sviluppo minimale a microprocessore richiede normalmente sei LED; quattro LED sono richiesti per visualizzare l'indirizzo (quattro digit esadecimali rappresentano 16 bit) e due LED vengono usati per i dati (otto bit). Il terzo dispositivo di uscita tipico è la stampante. Uno dei più economici ed affidabili dispositivi è stato ed è tuttora la telescrivente standard (TTY). Naturalmente sono disponibili molti altri dispositivi di output e il sistema a microprocessore può essere collegato a qualsiasi meccanismo di controllo esterno, a partire da un relè fino a un motore passo-passo.

Le due *memorie di massa* più frequentemente usate con i microprocessori sono il nastro in cassette e il floppy-disk, in particolare i nuovi mini o microfloppy. Il *nastro in cassette* è la memoria di massa più economica fra le disponibili. Utilizzando un registratore a nastro standard di tipo audio, un gran numero di programmi possono venir facilmente memorizzati su cassette standard. Lo svantaggio è che il registratore deve generalmente venir acceso e spento manualmente e anche il tasto di registrazione deve venir attivato manualmente.

Inoltre il registratore audio è lento e relativamente inaffidabile. Tuttavia per tutti i fini pratici esso è il mezzo ideale per memorizzare programmi nei sistemi molto piccoli; (è facile scrivere un programma due volte nel caso che la prima copia sia stata danneggiata). Il registratore è economico e la affidabilità è generalmente sufficiente per tutte le necessità usuali in una situazione non industriale. Però è lento. Per esempio un interpretatore base standard richiederà approssimativamente 30 secondi di tempo di svolgimento del nastro. L'interfaccia richiesta per collegarsi ad un registratore è semplicissima. Vengono eseguiti collegamenti diretti all'input del microfono e dall'output audio (jack esterno) mediante una semplice interfaccia.

Il dispositivo standard di memoria di massa per gli usi «professionali» dei microprocessori è il floppy-disk o il nuovo minifloppy. Il floppy-disk è un disco magnetico morbido. I bit sono registrati sulla superficie del disco usando una testina mobile. La superficie del disco viene divisa logicamente in tracce e settori. I settori corrispondono alle fette di una torta. Le tracce sono anelli concentrici. La dimensione tipica di un blocco di dati entro un settore di una traccia è di 128 parole. Un floppy-disk fornisce tipicamente 110K byte a 125K bit al secondo. Esso consuma da 7 a 15 Watt. Il meccanismo di trascinamento costa 200÷300 $. Richiede un controllore di disco per poter essere interfacciato al sistema microprocessore. Oggi questi controllori esistono in un singolo chip, forniti da diversi costruttori. La maggior parte dei sistemi hanno di fatto non solo uno, ma due floppy-disk. C'è una ragione concettuale ed una finanziaria per questo sdoppiamento. La ragione concettuale sta nel fatto che i floppy-disk sono usati tipicamente per preservare dei file. Se i file debbono venir prelevati o riuniti assieme, è necessario avere accesso a due file simultaneamente cioé usare due floppy-disk. Inoltre un sistema di trascinamento per due dischi è solamente poco più costoso che per un disco singolo e può utilizzare il medesimo controllore. Per questa ragione la maggior parte dei sistemi forniti di floppy disk usano il trascinamento duplice.

Si specula da più parti che le nuove tecnologie di memoria come la CCD e le memorie a bolle sostituiranno in futuro i dischi. Questo avverrà comunque tra diversi anni.

SOMMARIO

Ora sono stati presentati tutti i componenti e le funzioni richieste per allestire un sistema. Siamo quasi pronti per interconnetterli. Possono sorgere delle differenze in funzione del microprocessore scelto. Ora verranno studiate le caratteristiche e le differenze tra i microprocessori esistenti.

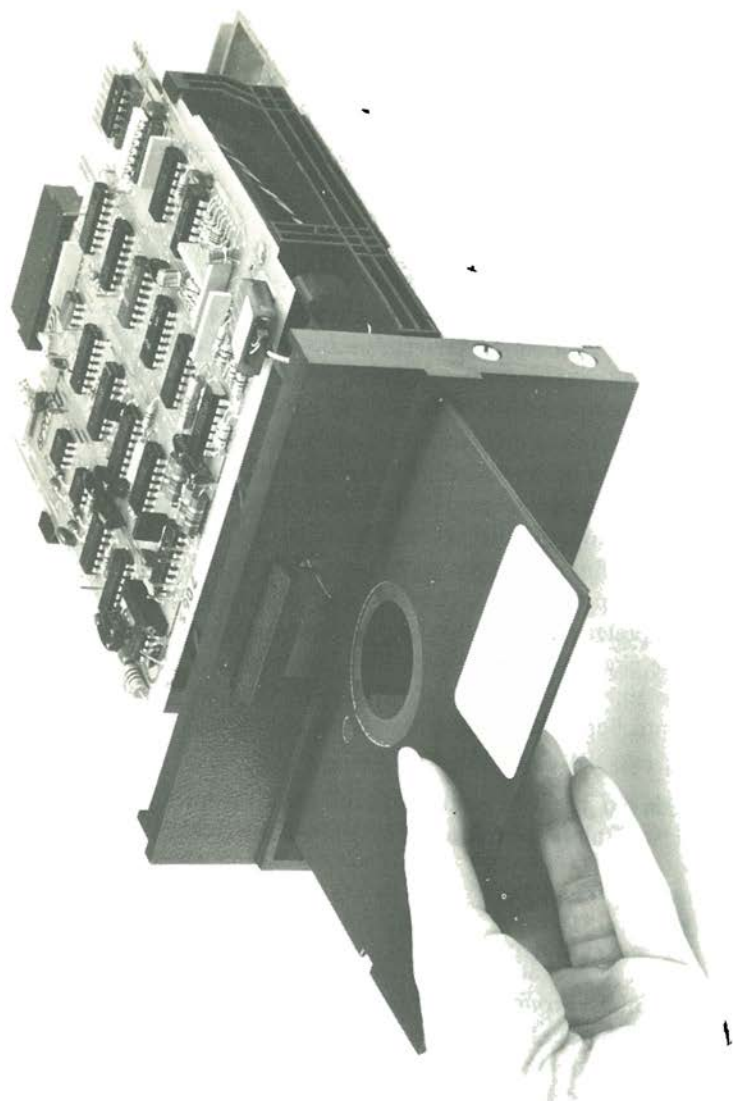



Fig. 3-44: Inserzione di un diskette in un *mini - floppy*.

CAPITOLO 4

VALUTAZIONE COMPARATIVA TRA MICROPROCESSORI

OBIETTIVO

L'obiettivo di questo capitolo è di fornire una panoramica comparativa sui microprocessori oggi sul mercato. Verrà posta un'attenzione speciale sulle caratteristiche che qualificano ciascun prodotto per applicazioni specifiche. Verranno analizzati i vantaggi e gli svantaggi rispettivi di ciascun microprocessore. Dopo aver stabilito queste caratteristiche, verranno presentati nella seconda parte del capitolo i criteri essenziali per la scelta di un microprocessore.

ELEMENTI FUNZIONALI DI UNA MPU

Si possono usare diversi criteri di scelta per differenziare i diversi microprocessori oggi sul mercato. Uno degli aspetti fondamentali nella valutazione tecnica di un microprocessore è quale livello di complessità è stato integrato su un singolo chip.

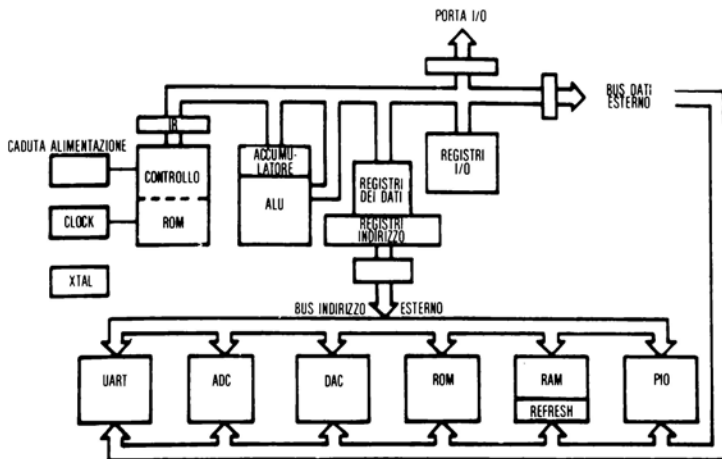



Fig. 4-1: Moduli funzionali di un sistema microprocessore.

L'insieme delle funzioni necessarie per il funzionamento di un sistema completo è stato presentato dal Capitolo 1 al Capitolo 3. I moduli effettivi che dovrebbero essere disponibili in un sistema completo compaiono in Fig. 4-1. La metà superiore

della figura rappresenta essenzialmente le funzioni incorporate in un chip di micro processore standard. La metà inferiore della figura rappresenta le differenti funzioni che usualmente devono venir aggiunte al sistema. In un sistema standard esse sono fornite da chip separati.

Osservando per prima cosa la parte superiore della figura, ogni microprocessore deve includere almeno un' unità di controllo, una ALU e dei registri. L'unità di controllo praticamente di tutti i microprocessori viene realizzata mediante un microprogramma. Ciò spiega la presenza di una ROM nel modulo della unità di controllo in figura. Il vantaggio di usare una ROM per la sequenzializzazione della «control - unit» è duplice:

1. Aumenta l'utilizzazione dell'area limitata del chip fornendo più elaborazione logica di quanta se ne otterrebbe usando una progettazione convenzionale di logica distribuita.

2. Essendo programmata, la CU può essere prontamente modificata. Questo è un vantaggio fondamentale per correggere gli errori iniziali di progettazione e per aggiungere funzioni e miglioramenti in un secondo tempo.

La ALU in figura è fornita di un'accumulatore poiché questa è l'architettura standard dei microprocessori d'oggi. Successivamente, spostandoci a destra, i registri dei dati sono da 8 bit e i registri degli indirizzi sono da 16 bit. Essi includono registri come il PC, l'SP e qualsiasi altro puntatore di indirizzo addizionale che possa essere stato previsto (l'IX per esempio). Sulla destra della figura compare un gruppo di registri di I/O. Si è iniziato ad utilizzarli nei progetti recenti per poter fornire migliori possibilità di input-output. Questi registri non sono disponibili sui microprocessori più diffusi. Essi vengono sempre previsti quando la memoria è incorporata nel chip, lasciando liberi con ciò un certo numero di piedini per l'input e per l'output (i piedini che sono stati lasciati disponibili dal precedente address - bus).

Nella parte inferiore della figura compaiono molti dei chip più comuni necessari per completare il progetto del sistema: la ROM e la RAM qualora esse non siano incluse nel chip medesimo: la UART per le comunicazioni seriali; il PIO per le comunicazioni in parallelo; l'ADC e il DAC per le conversioni analogico/digitali e digitali/analogiche. Inoltre un sistema complesso può richiedere un PIC (controllore di interrupt programmato), un PIT (temporizzatore di intervalli programmabile) o un DMA (accesso diretto in memoria) per una efficiente gestione dell'I/O.

Infine nella parte in alto a sinistra della figura compaiono due moduli che oggi sono quasi sempre integrati nel chip medesimo: il circuito oscillatore (clock) e il circuito di restart in caso di interruzione dell'alimentazione «power - fail - restart» (PFR).

Un modo semplice per caratterizzare le funzioni realizzate su qualunque chip di microprocessore consiste nell'esaminare la Fig. 4-1 e nel circoscrivere i moduli disponibili entro il microprocessore. Per assiemare un sistema, qualsiasi modulo non

fornito entro lo stesso chip del microprocessore dovrà essere inserito mediante chip esterni.

Ad esempio un 8080 o un 6800 metteranno a disposizione essenzialmente le funzioni della parte alta della figura eccetto il clock, il power fail e i registri di I/O. Ne segue che sarà necessario aggiungere la ROM, la RAM, la UART, il PIO più qualsiasi altro chip specializzato che possa essere necessario.

Un microcomputer in un singolo chip, come un 8048 o un 3870, fornirà su un singolo chip, tutte le possibilità della parte superiore sinistra della figura più gran parte dei moduli che compaiono nella parte inferiore: ROM, RAM e PIT.

Fig. 4-2: Evoluzione dei microprocessori.

Si può pronosticare che i nuovi microcomputer che appariranno in futuro incorporeranno infine queste funzioni entro un singolo chip. Allora non saranno più necessari chip esterni per realizzare un sistema completo. Si può anche prevedere che appariranno sistemi molto più complessi. Questi sistemi complessi si potranno ottenere creando chip che realizzino combinazioni di questi moduli. In particolare realizzando un chip che fornisca la gestione dell'I/O in aggiunta ad un processore si potranno eventualmente creare sistemi potenti multiprocessori. Queste nuove inter-

facce di I/O munite di processore dovrebbero iniziare ad apparire entro un breve tempo.

CLASSIFICAZIONE DEI MICROPROCESSORI

Sono possibili diverse classificazioni funzionali. In termini di architettura sono state già presentate le quattro classi: microprocessore standard, microcomputer su un chip, sistema a due chip, bit-slice.

Qui verrà usata una classificazione differente, più adatta a paragoni mediante tabelle:

1. microprocessori a 4 bit;
2. microprocessori a 8 (o 12) bit;
3. microprocessori a 16 bit
4. bit-slice

In ciascun paragrafo verrà presentata una tabella riassuntiva che elencherà le caratteristiche tecniche di ciascun microprocessore in un formato adatto ai paragoni. La tabella per i microprocessori a 4 bit compare in Fig. 4-4. I termini usati nella tabella hanno i seguenti significati:

- standard: rappresenta il tipo di architettura: quindi parlare di architettura standard a microprocessore significa una CPU su un chip con tre bus: per i dati, gli indirizzi, il controllo.
- μC: rappresenta una architettura microcomputer sia essa su uno o due chip (per semplificare). È un microprocessore che incorpora la CPU e la memoria (ROM-RAM).
- tempo di ciclo: rappresenta il tempo tipico di esecuzione di una istruzione e non il tempo di ciclo del clock.
- numero di istruzioni - è in genere il termine più facilmente mal interpretato. È possibile elencare il numero di istruzioni di qualsiasi processore e poi moltiplicarlo facilmente per una potenza del 2. Basta elencare separatamente le istruzioni che agiscono sul registro A, poi sul registro B, poi sul registro C. In funzione delle convenzioni scelte il numero delle istruzioni può allora essere modificato a piacere. Per questa ragione si ha la sensazione che il numero delle istruzioni non sia significativo. Il numero delle istruzioni per i microprocessori a 8 bit è essenzialmente limitato dalla necessità di fornire istruzioni di 1 byte, lasciando liberi 6 bit per l'opcode. Ciò rappresenta di fatto 64 opcode differenti.
- PMOS e NMOS: si riferisce alla tecnologia utilizzata
- stack: «hard» indica una realizzazione hardware dello stack, cioé un gruppo di registri hardware interni inseriti entro la MPU per le operazioni di stack. «Soft» indica una realizzazione software dello stack cioé la disponibilità di uno stack-pointer che punta verso l'inizio dell'area di stack entro la memoria.
- interrupts: denota la disponibilità di interrupts e quanti livelli sono previsti. Nel caso non fossero disponibili questo è indicato mediante uno «0» o un «no».

- le altre indicazioni si spiegano da sole.

Rimarchiamo che i numeri presentati in questa tabella sono stati ricercati con cura. Tuttavia potrebbero essere stati fatti degli errori nella tabulazione oppure potrebbero nel frattempo essere stati introdotti nuovi prodotti che potrebbero aver reso obsoleti alcuni dei rilievi. Per questa ragione è decisamente raccomandabile che il lettore si riferisca alla documentazione corrente del costruttore per le caratteristiche effettive.

MICROPROCESSORI A 4 BIT

I microprocessori a 4 bit sono meno complessi di un progetto ad 8 bit e possono venir prodotti in linea di principio con una miglior resa. Ciò conduce ad un minor costo per questi componenti. Nel campo dei microcomputer in un chip il costo è un criterio molto significativo. I microcomputer a 4 bit dovrebbero perciò essere competitivi per alcuni anni a venire. Eventualmente essi verranno sostituiti da progetti a 8 bit e in un più lontano futuro ci si può attendere il dominio del progetto a 16 bit. In futuro diverranno senza dubbio disponibili progetti con parola ancora più «ampia».

Intel

L'Intel 4004 è il nonno dei microprocessori. Fu progettato originariamente come elemento di elaborazione di una calcolatrice da tavolo per un costruttore Giapponese e fu introdotto nel lontano 1971. Il 4004 non fu mai progettato come computer di impiego generale. Le sue mancanze furono riconosciute non appena fu presentato. Tuttavia esso fu il primo dispositivo di elaborazione di impiego generale di un chip ad essere posto sul mercato. Molti chip presentati quasi contemporaneamente da altre ditte e chiamati «microprocessori» erano, in realtà, semplici chip per calcolatori. Alcuni di essi erano persino dispositivi a bit seriali.

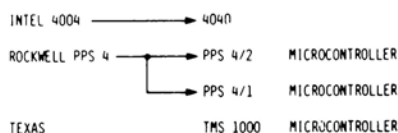



Fig. 4-3: Microprocessori a 4 bit.

Oggi il 4004 è obsoleto ed è stato sostituito dal 4040. Tuttavia vengono ancora presentati dispositivi che usano il 4004 in numero significativo. Come è possibile ciò? La ragione è semplice. Per educare la comunità di tecnici e progettisti e poi del management c'è voluto un tempo molto lungo. Una volta che il management si decideva ad accettare l'uso di un microprocessore, ciò non aveva alcun valore, nell'ambito di molte società, come spinta verso l'accettazione di un successivo processore.

Il 4004 fu rapidamente rimpiazzato da una versione migliorata, il 4040. Per preservare gli investimenti di sviluppo di software degli utilizzatori del 4004, il set di istruzioni del 4040 è compatibile con quello del 4004. Il 4040 offre miglioramenti significativi. In particolare esso è fornito di capacità di interrupt e di un gran numero di registri interni. È equipaggiato con tre banchi di registri (3x8 registri). La disponibilità di tre banchi di registri permette, in linea di principio, dei responsi agli interrupt molto veloci. Non appena avviene una richiesta di interrupt, purché non vi siano più di due livelli di interrupt, si deve semplicemente commutare da un banco di registri all'altro. Sfortunatamente il program counter e lo status word non sono replicati entro i tre banchi di registri. Questo richiede un certo numero di istruzioni software da eseguire per una commutazione completa.

	INTEL 4004	INTEL 4040	ROCKWELL PPS4	ROCKWELL PPS4/1	ROCKWELL PPS4/2	TMS 1000
TIPO	STANDARD	STANDARD	STANDARD	uC	2-CHIP uC	uC
TECNOLOGIA	PMOS	PMOS	PMOS	PMOS	PMOS	PMOS
NUMERO ISTRUZIONI	46	60	50	50	50	PICCOLO
TEMPO DI CICLO (μS)	10,8	10,8	5	12,5	5	15
INDIRIZZAMENTO DIRETTO (BIT)	12	12	12	11	11	10 o 11
REGISTRI	16	24	4	3 + RAM	2 + RAM	4
STACK (LIVELLI)	HARD (3)	HARD (7)	HARD (1)	HARD (2)	--	HARD (1)
INTERRUPTS	Ø	1	1	2	NO	--
CLOCK SUL CHIP	--	--	--	--	si	si
ROM SUL CHIP (BYTES)	--	--	--	1K	--	2K
RAM SUL CHIP (PAROLE)	--	--	--	96	--	128
TEMPORIZZATORI SUL CHIP	--	--	--	--	--	--
FFR	--	--	--	--	--	--
ALTRI	--	--	--	--	--	--
LINEE I/O	--	--	12	26	12	19 a 24
CONTENITORE (PINS)	16	16, 24	42	42	42	40 o 28
ALIMENTAZIONE (V)	15	15	17	15	15	15
NOTE		SET DI IST COMPATIBILE CON 4004	BCD ADD IN 30 uS		PPS 4 INSTRN SET	

Fig. 4-4: Tabella comparativa dei microprocessori a 4 bit.

Le prerogative del 4040 sono limitate, ma sufficienti per molte applicazioni semplici. Di conseguenza è stato incorporato in un grande numero di applicazioni industriali. Tuttavia poiché oggi è possibile acquistare un processore a 8 bit praticamente al medesimo prezzo del 4040, la porzione di mercato di quest'ultimo si ridurrà progressivamente.

Anche con le limitazioni indicate si può prevedere che il 4040 continuerà ancora ad essere usato per diverso tempo nel mercato dei ricambi. Una volta che un microprocessore è stato impiegato per un progetto, purché le sue prestazioni siano sufficienti, non c'è nessun beneficio diretto a sostituirlo con un altro microprocessore.

Le eventuali funzioni aggiuntive verranno ottenute tramite programmazione. Questa è una delle capacità tipiche di un microprocessore: un progetto hardware standardizzato e modifiche e cambiamenti successivi ottenuti modificando il software. Il costo di riprogettazione di una scheda o di un prodotto industriale completo comprendono il progetto meccanico, la documentazione, il training, la documentazione ecc, è enorme. Tuttavia molto probabilmente la maggior parte dei nuovi progetti impiegherà microprocessori a 8 bit.

Rockwell

Nel 1971 la Rockwell aveva in produzione diversi chip per calcolatore. Seguendo un'evoluzione simile ai prodotti Intel, fu introdotto il PPS4 come evoluzione di un progetto per calcolatore. Tuttavia fu introdotto in ritardo sul mercato e, nell'attesa che divenisse adeguatamente supportato, i progetti con microprocessore a 8 bit avevano invaso il campo. Il successore del PPS4 è il PPS4/2, un calcolatore completo su due chip. Il set di istruzioni del PPS4/2 è compatibile con il PPS4. Esso ha un concorrente diretto nell'Intel 4040 nei progetti semplici e a basso costo.

Il prodotto più interessante è il nuovo PPS4/1, uno dei primi completi microcomputer su un solo chip, introdotto nel 1976. È un progetto potente con un gran numero di linee di input e di output, che incorpora ROM, RAM e clock direttamente sul chip. Esiste in diverse versioni in funzione della quantità di memoria incorporata sul chip e del numero di pin. Il prezzo per un tale dispositivo in grandi quantitativi è ben al di sotto dei 5 $. Le applicazioni tipiche sono indirizzate ai beni di consumo che giustificano il costo di produrre molte migliaia di unità con una ROM mascherata. La PPS4/1 risente ora di una forte concorrenza da parte di nuovi microcomputer su un solo chip che verranno descritti nel paragrafo seguente. I suoi due vantaggi principali sono: disponibilità completa e basso costo. Esso è in concorrenza anche con il TMS1000 della Texas Instruments.

Texas Instruments

La Texas Instruments ha tardato ad entrare nel campo dei microprocessori. Ha introdotto direttamente la famiglia TM1000, un competo microcomputer su un unico chip a 4 bit. La famiglia TMS1000 incorpora ROM, RAM e circuito di clock in un unico chip. È prevista per applicazioni di largo consumo e per prodotti industriali di grande serie. Tuttavia la sua architettura interna è significativamente diversa dal progetto standard di elaboratore descritto fino ad ora. Il TMS1000 fornisce un certo numero di segnali di controllo decodificati internamente mediante un PLA. Il vantaggio di usare un PLA interno è duplice: 1) l'applicazione dell'utilizzatore è codificata in un modo che è difficile da decifrare e perciò da copiare da parte della concorrenza; 2) sostituisce una quantità molto maggiore di ROM per fornire segnali di uscita in risposta a combinazioni di segnali di ingresso. Tuttavia il TMS1000 non ha l'abbondanza di porte dirette di I/O a 4 bit del PPS4/1. Esso è perciò particolarmente adatto a quelle applicazioni ove è necessario un controllo e-

laborato in risposta alle «istruzioni» di 4 bit. Non è particolarmente adatto all'elaborazione diretta di flussi di dati di 4 bit. Come per il PPS4/1, il prezzo del TMS1000 per grandi quantitativi è molto basso (in genere i costruttori forniscono le quotazioni solo su richiesta).

MICROCOMPUTER SU 1 CHIP A 4 BIT

I due microcomputer su 1 chip a 4 bit sul mercato oggi sono il Rockwell PPS4/1 e la famiglia Texas TMS1000 (entrambi hanno molti membri appartenenti alla fa-

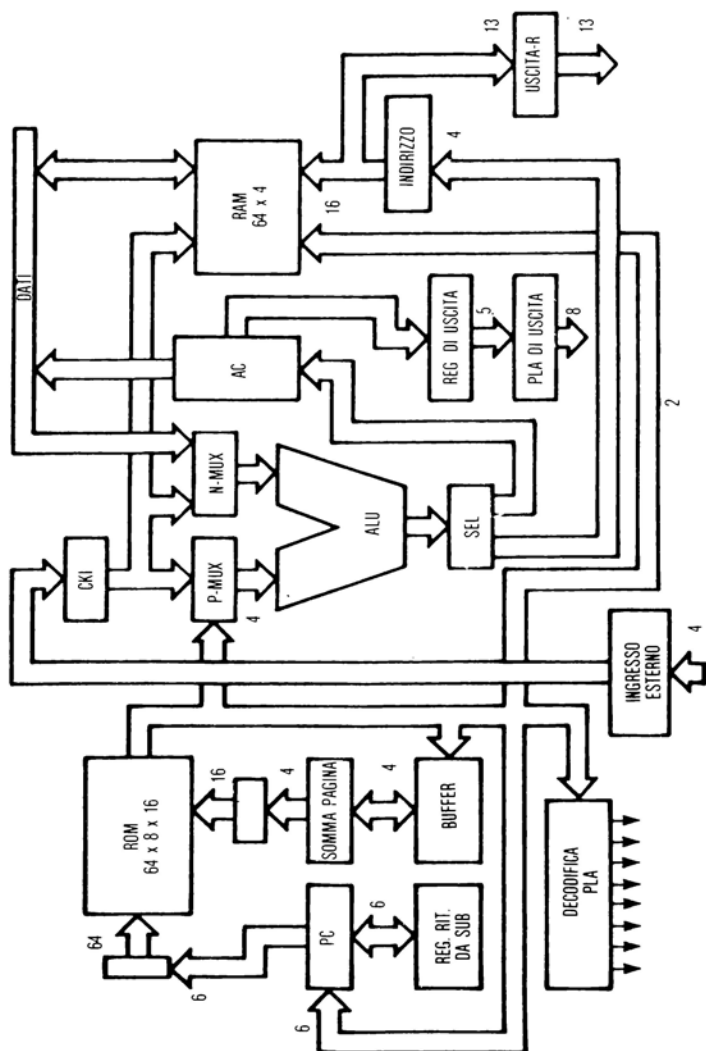



Fig. 4-5: TMS1000 della Texas.

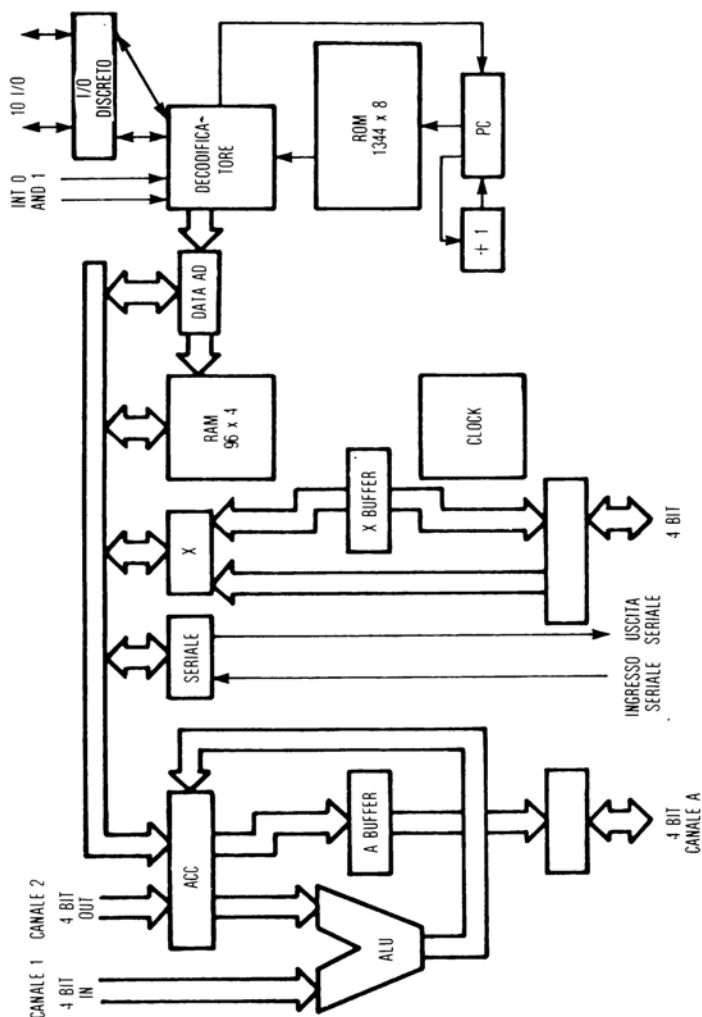



Fig. 4-6: PPS4/1 della Rockwell.

miglia, in funzione della quantità di memoria inclusa nel chip e del numero di piedini). Entrambi i dispositivi usano ROM programmate con mascheratura. Per considerarle economiche è necessario un ordine minimo di diverse migliaia di dispositivi. Infatti in genere si considerano quantità minime di almeno 10.000 unità. Essi non sono adatti all'utilizzatore su piccola scala. Il loro costo in grandi quantità è un criterio essenziale di scelta. Il supporto è naturalmente il secondo criterio, cioé la facilità con cui essi possono essere usati e programmati. Entrambi questi chip sono stati ormai prodotti in grandi quantitativi e possiedono un'affidabilità accertata.

Ora è diventato possibile creare un microcomputer su 1 chip a 8 bit. I progetti esistenti verranno descritti nel prossimo paragrafo. Non appena il prezzo dei micro-

computer a 8 bit sarà sceso ad un livello simile al prezzo di quelli a 4 bit è prevedibile che i microcomputer a 8 bit domineranno il campo. Comunque è stato precisato che generalmente il costo è il parametro più significativo nella scelta di un microcomputer su 1 chip. Per questa ragione ci si può aspettare che i progetti a 4 bit manterranno ancora per diverso tempo una porzione significativa del mercato. Eventualmente questa diminuirà con una evoluzione simile alla «standardizzazione» del microprocessore a 8 bit.

È improbabile che venga introdotto ora qualsiasi nuovo progetto nel campo dei 4 bit. I vantaggi ovvi dei progetti a 8 bit e a 16 bit sono una maggior velocità di elaborazione e, spesso, la compatibilità con i set di istruzioni o le architetture esistenti. La tecnologia rende ora possibile considerare progetti ancor più complessi. Poichè il microcomputer su 1 chip è orientato alle applicazioni che presuppongono grandissimi quantitativi, un costruttore che voglia realizzare vendite di diverse centinaia di migliaia di unità potrà abbassare i suoi prezzi ad un livello inferiore a quello dei progetti a 4 bit più venduti. Per questa ragione c'è l'incentivo a sviluppare il progetto più complesso che il costruttore sia in grado di vendere in grandissimi quantitativi.

MICROPROCESSORI A 8 BIT

È stato indicato che un'importante limitazione degli odierni microprocessori è la limitazione dei pin del package a 40. A causa di questa limitazione i microprocessori a 16 bit non possono comunicare con il mondo esterno senza multiplexare l'informazione. I microprocessori a 8 bit sono diventati la soluzione standard sul mer-

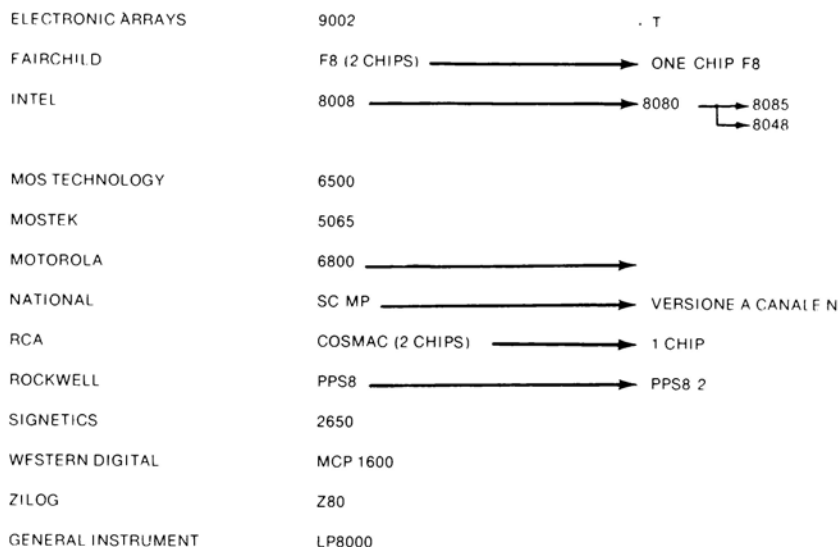



Fig. 4-7: Microprocessori ad 8 bit.

cato. Fino all'avvento dei microcomputer su 1 chip, il microprocessore standard su un chip è stato quello più frequentemente usato su ogni tipo di applicazione (fino alla fine del 1976). Un microprocessore a 8 bit fornisce tipicamente una capacità di elaborazione significativamente più potente di un progetto a 4 bit, ad un costo simile. Esso ha superato il progetto a 4 bit.

L'elenco dei microprocessori esistenti a 8 bit compare in Fig. 4-7. L'evoluzione di questi prodotti verrà ricordata e le loro caratteristiche comparate, costruttore per costruttore. La lista è divisa in due parti fondamentali: i microprocessori standard a 8 bit e i nuovi microcomputer su un unico chip a 8 bit. Per primi verranno esaminati i microprocessori standard a 8 bit.

	INTEL 8008	INTEL 8080	INTEL 8085	INTEL 8048
TIPO	STANDARD	STANDARD	STANDARD	uC
TECNOLOGIA	PMOS	NMOS	NMOS	NMOS
NUMERO ISTRUZIONI	48	69	71	50
TEMPO DI CICLO (μS)	12.5, 20.0	1.3, 1.5, 2.0	1.3	2.5
INDIRIZZAMENTO DIRETTO (BIT)	14	16	8 + 8	--
REGISTRI	7	7	7	1
STACK (LIVELLI)	HARD (7)	SOFT	SOFT	HARD (8)
INTERRUPTS	1	1	4	1
CLOCK SUL CHIP	--	--	si	si
ROM SUL CHIP (BYTES)	--	--	--	1K
RAM SUL CHIP (PAROLE)	--	--	--	64
TEMPORIZZATORI SUL CHIP	--	--	--	si
FFR	--	--	--	--
ALTRI	--	--	8228	--
LINEE I/O	--	--	2	27
CONTENITORE (PINS)	18	40	40	40
ALIMENTAZIONE (V)	-9, +5	+5, -5, +12	5	5
SECONDA SORGENTE	--	AMD, TI, NEC, NS, SIEMENS	--	--
NOTE	--	--	--	--

Fig. 4-8: Tabella comparativa dei microprocessori ad 8 bit.

Intel

Il precursore dei microprocessori a 8 bit è l'Intel 8008 introdotto nel 1972-1973. L'8008 non si proponeva di essere un microprocessore di impiego generale. Era stato concepito come controllore per un visualizzatore CRT per la Datapoint. Le condizioni che hanno circondato la sua introduzione sul mercato sono state già descritte al capitolo 1. Nonostante tutte le sue inadeguatezze di progetto e le possibilità limitate, l'8008 ebbe un successo strepitoso. Questo successo convinse un certo numero dei principali costruttori di semiconduttori a sviluppare dei progetti compe-

	EA 9002	FAIRCHILD F8 (2 CHIPS)	F8 (1-CHIP 3859)	MOS TECH 65XX	GI LP8000
TIPO	STANDARD	uC	uC	STANDARD	uC (2-CHIP)
TECNOLOGIA	NMOS	NMOS	NMOS	NMOS	--
NUMERO ISTRUZIONI	55	70	70	come 6800	--
TEMPO DI CICLO (μS)	2 a 4	2 a 13	1.5	--	--
INDIRIZZAMENTO DIRETTO (BIT)	12	16	--	12 a 16	11
REGISTRI	8	1 + RAM	1 + RAM	3	RAM + AC
STACK (LIVELLI)	HARD (7)	EXTERNAL SP	SOFT	SOFT (8 BITS)	HARD (4)
INTERRUPTS	1	1	1	1	--
CLOCK SUL CHIP	si	--	si	si	--
ROM SUL CHIP (BYTES)	--	--	1 K	--	--
RAM SUL CHIP (PAROLE)	64	64	64	--	48
TEMPORIZZATORI SUL CHIP	--	--	YES	--	--
FFR	--	--	--	--	--
ALTRI	--	--	--	--	--
LINEE I/O	--	16	32	--	8
CONTENITORE (PINS)	28	40	40	28/40	--
ALIMENTAZIONE (V)	5	+5, +12	+5, +12	--	--
SECONDA SORGENTE	--	MOSTEK	MOSTEK	SYNERTEK	--
NOTE	--	--	--	--	--

Fig. 4-9: Tabella comparativa dei microprocessori ad 8 bit (segue).

titivi. L'8080 fu progettato come successore dell'8008 col quale doveva mantenere la compatibilità. Per mantenere la compatibilità esso include tutti i registri dell'8008, più alcuni altri e tutte le istruzioni dell'8008, più alcune altre. L'8080 fu il primo microprocessore potente introdotto sul mercato e tuttora è il più venduto nella propria classe. Diversi altri microprocessori con caratteristiche simili sarebbero stati introdotti sul mercato l'anno successivo o più tardi. Tecnicamente l'8080 non è il prodotto migliore sul mercato. Tuttavia fino ad ora l'8080 ha avuto il successo di vendita più alto di ogni altro microprocessore standard. Ciò è dovuto a diversi fattori, uno dei quali sta certamente nel fatto che fu introdotto per primo. La seconda ragione probabilmente sta nel fatto che l'Intel fu tra le prime società ad investire nello sviluppo di chip di supporto e di software di supporto per i propri prodotti. Tutti i concorrenti dell'8080 furono presentati con un ritardo di nove mesi (Motorola 6800) e più. L'8080 ha ora un «successore», l'8085. Questo prodotto verrà descritto più avanti in questo capitolo, una volta che i termini di paragone saranno stati stabiliti. Abbandoniamo per il momento la Intel.

Motorola

La Motorola ha introdotto il 6800 come concorrente diretto all'8080. Il progetto del 6800 è stato ovviamente ispirato dall'8008 e dalla filosofia dei minicomputer. Il 6800 ha essenzialmente la medesima architettura interna dell'8080. Ha alcune differenze a livello dei registri. Il 6800 è fornito di due accumulatori contro uno solo per l'8080, ma possiede un numero ridotto di registri di impiego generale. Il 6800

ha un IX speciale, o registro indice, che facilita l'accesso a tabelle conservate in memoria. L'8080 non ha registro indice, ma è fornito di coppie di registri che possono essere usati per fornire possibilità simili. Le istruzioni del 6800 riflettono probabilmente il fatto che questo fu introdotto successivamente all'8080. Dette istruzioni sembrano essere un po' più complicate ma in generale sono simili a quelle dell'8080. In genere può risultare marginalmente più veloce l'uno o l'altro microprocessore come conseguenza delle funzioni utilizzate nel paragone. Le differenze più significative di rendimento si possono ottenere, escludendo il paragone tra un 8080 standard e un 6800 standard, (il rispettivo rendimento è essenzialmente simile), acquistando semplicemente una versione più veloce dell'8080 o del 6800.

L'8080 è disponibile in tre versioni, lo standard 8080A con il clock a 2 MHz, l'8080 A-2 e l'8080 A-1 con il clock a 3 MHz. Anche il 6800 è disponibile ora in due versioni: il 6800 standard usa un clock di 1 MHz. Questo fatto dovrebbe forse indicare che l'8080A standard ha velocità doppia del 6800 standard? No. Il clock fornisce semplicemente gli impulsi necessari dal microprogramma interno della control unit. L'8080 in media richiede un numero doppio di microistruzioni di quello richiesto dal 6800. La loro funzionalità è mediamente simile. Una instruzione «tipica» dell'8080 viene eseguita in due microsecondi e la maggior parte delle istruzioni del 6800 viene eseguita in due microsecondi. Un paragone più dettagliato risulterebbe non conclusivo.

Un criterio di paragone di base potrebbe essere l'esecuzione di un «programma standard» sui microprocessori da paragonare. Purtroppo però non esiste alcun pro-

	MOSTEK 5065	MOTOROLA 6800	NS SC/MP	RCA COSMAC	SIGNETICS 2650
TIPO	STANDARD	STANDARD	STANDARD	.STANDARD	STANDARD
TECNOLOGIA	PMOS	NMOS	PMOS	CMOS	NMOS
NUMERO ISTRUZIONI	51	71	50	59	75
TEMPO DI CICLO (μS)	10	2	5 to 25	3	4.8 to 9.6
INDIRIZZAMENTO DIRETTO (BIT)	15	16	12	16	15
REGISTRI	3	4	6	16 (RAM)	7
STACK (LIVELLI)	--	SOFT	SOFT	NO	HARD (8)
INTERRUPTS	2	1	1	1	1
CLOCK SUL CHIP	--	--	si	si	--
ROM SUL CHIP (BYTES)	--	--	--	--	--
RAM SUL CHIP (PAROLE)	--	--	--	si	--
TEMPORIZZATORI SUL CHIP	--	--	--	--	--
FFR	--	--	--	-	--
ALTRI	--	--	--	--	--
LINEE I/O	--	--	2	--	--
CONTENITORE (PINS)	40	40	40	40	40
ALIMENTAZIONE (V)	-12, -5, +5	5	+5, -7	3 a 10	5
SECONDA SORGENTE	--	AMI, SESCOSEM	--	--	--
NOTE	--	--	--	BASSO CONSUMO (50 mW)	SOMMATORE STATICO DI INDIRIZZO

Fig. 4-10: Tabella comparativa dei microprocessori ad 8 bit (segue).

	ROCKWELL PPS 8	ROCKWELL PPS 8/2	WEST DIG MCP1600	ZILOG Z80	INTERSIL IM 6100
TIPO	STANDARD	uC	MICROPROGRAMMED	STANDARD	STD, 12 BIT
TECNOLOGIA	PMOS	PMOS	NMOS	NMOS	CMOS
NUMERO ISTRUZIONI	90	109	--	8080 + MANY	PDP8
TEMPO DI CICLO (μS)	4	5	--	1.6	2.5 a 11.0
INDIRIZZAMENTO DIRETTO (BIT)	15	14	16	(16)	12
REGISTRI	3	3	--	17	3
STACK (LIVELLI)	SOFT (5-BIT SP)	SOFT (5-BIT)	--	SOFT	--
INTERRUPTS	3	3	--	3	1
CLOCK SUL CHIP	--	si	--	si	si
ROM SUL CHIP (BYTES)	--	--	--	--	--
RAM SUL CHIP (PAROLE)	--	--	--	--	--
TEMPORIZZATORI SUL CHIP	--	--	--	--	--
FFR	si	si	--	--	--
ALTRI	--	--	--	INTERNAL 8228 + MEM REFRESH	--
LINEE I/O	--	--	--	--	--
CONTENITORE (PINS)	42	42	--	40	40, 28
ALIMENTAZIONE (V)	17	17	--	5	5
SECONDA SORGENTE	--	--	--	MOSTEK	HARRIS
NOTE	BCD ADD = 12 uS	--	SET ISTR PDP 11	SET ISTR 8080	SET ISTR PDP 8 STATICO

Fig. 4-11: Tabella comparativa dei microprocessori ad 8 bit (segue).

gramma standard di paragone. Paragonando l'8080 e il 6800, una moltiplicazione di 8 bit per 8 bit può risultare più veloce su uno dei processori, mentre una moltiplicazione di 8 bit per 16 bit può risultare eseguibile più velocemente sull'altro, a causa di «caratteristiche speciali» (di fatto errori di progetto o peculiarità del progetto).

L'8080 ha due svantaggi evidenti se paragonato al 6800:

1. L'8080 richiede tre livelli di tensione d'alimentazione: +5V, -5V, +12V. Poichè l'8080 fu introdotto ai tempi in cui l'Intel utilizzava la propria tecnologia per le memorie dinamiche, si stimò che i tre livelli fossero necessari. Il 6800 per contrasto richiede un unico livello di alimentazione. La differenza principale sta nel fatto che ciò lascia liberi due pin sul package e che esso richiede un alimentatore più semplice. L'argomento di un'alimentazione più semplice potrebbe non essere valido: se il sistema utilizza una memoria dinamica richiede generalmente in ogni caso i tre livelli +5V, -5V e +12V sulla scheda.

2. La mancanza di pin ha condotto ad un altro svantaggio: per mancanza di piedini l'8080 deve usare il data bus per inviare l'informazione di stato durante lo stato T1 di ciascun ciclo macchina. È necessario poi multiplexare esternamente il bus dei dati. Ciò si realizza con un chip in più, l'8228 system controller. Quando l'8080 era stato introdotto da poco questo chip non esisteva e detto punto era decisamente uno svantaggio. Tuttavia venne introdotto ben presto l'8228 e verrà dimostrato come ciò non si traduca in uno svantaggio. Nella maggior parte delle applicazioni l'8080 non richiede un componente in più se paragonato al 6800. È vero

che l'8080 richiede l'8228; tuttavia l'8228 non fornisce solamente il demultiplexing del data bus, ma anche un driver bidirezionale per il data bus allo stesso tempo. Un sistema costruito con il 6800 o l'8080 generalmente non è un sistema orientato alla minima complessità, cioè da richiedere un numero minimo di chip. La maggior parte dei sistemi che usano l'8080 o il 6800 necessitano di pilotare più di un carico TTL. Essi devono usare dei drivers sia per l'addres bus che per il data bus. L'8080 non ha bisogno di driver aggiuntivi per il proprio data bus. Il 6800 invece ne ha bisogno. Chip per chip, il totale dei componenti è il medesimo. Entrambi i sistemi richiedono in più, naturalmente, un oscillatore esterno (un clock, più un cristallo di quarzo).

La Motorola non è stata l'unico costruttore a puntare sul nuovo mercato dei microprocessori introducendo un concorrente all'8080. (Il 6800 è stato il secondo microprocessore standard come volume di vendita nel 1976). La maggior parte dei costruttori si è impegnata in questo campo. Di fatto alcuni costruttori come la MOS Technology hanno realizzato un concorrente diretto del 6800 (la famiglia 6500 che verrà descritta più avanti). Esaminiamo ora altri concorrenti dell'8080.

Fornitori Alternativi

In questo settore industriale si possono distinguere due tipi di fornitori alternativi per un microprocessore: fornitori alternativi (second source) autorizzati e fornitori alternativi non autorizzati. Il risultato è il medesimo; un altro costruttore produce un componente che dovrebbe essere identico al modello. Ciò può ridurre le vendite del primo componente. Tuttavia si genera così una stabilità di mercato richiesta dalla maggior parte dei clienti industriali e militari e in genere contribuisce alla pubblicizzazione di un chip. Lo scambio incrociato dei brevetti è diventato molto vantaggioso in questi ultimi tempi. Naturalmente i microprocessori più venduti sono stati copiati da molti costruttori sia per profitto, sia per acquisire esperienza nella tecnologia e nel progetto del processore. Le copie non autorizzate sono anche chiamate spesso versioni allo «scanning electron microscope» (cioè realizzate con il microscopio elettronico a scansione). Ad esempio l'8080 ha solamente un fornitore alternativo autorizzato fino ad ora (anche se altre società hanno stipulato degli accordi che le autorizza a produrre l'8080 se fosse necessario): questa è la Siemens in Germania. Questo costruttore possiede le maschere e il processo per costruire il chip e li usa. Ma la maggior parte dei fornitori alternativi dell'8080 non è autorizzata. Essi sono, per esempio: l'AMD9080 della AMD, una versione completamente compatibile con l'8080 che viene pubblicizzata come più veloce e un po' più affidabile. Esistono poi l'8080 della Texas Instruments, della NEC (in Giappone) e della National Semiconductor. La maggior parte di questi componenti viene pubblicizzata come totalmente compatibile con l'8080, sia come compatibilità pin per pin, sia come compatibilità di software. Alcune delle prime versioni non erano completamente compatibili e ciò causò alcune sorprese agli utilizzatori.

L'impatto dei fornitori alternativi sul mercato può essere molto significativo. Si

può attribuire probabilmente alla AMD il merito di aver iniziato la più grossa guerra dei prezzi sui microprocessori. Ciò determinò una caduta di prezzi dell'8080 e di prodotti similari che è stato uno degli eventi più spettacolari del 1975. Ai tempi in cui la Intel vendeva l'8080 per 70 $ (sempre in quantità da 100 pezzi in su), la AMD annunciò il 9080 per 35 $. Entro pochi giorni l'Intel annunciava l'8080 a 35 $. Oggi il 9080 della AMD è quotato a 12 $ (forse anche meno per la data in cui leggerete questo libro). L'Intel non ha seguito la nuova riduzione o forse non ancora. Naturalmente questo ha costretto gli altri costruttori in concorrenza con il medesimo chip a riallineare i prezzi rispettivi. Da allora altri costruttori e in particolare la Texas Instruments sono entrati allegramente nella guerra dei prezzi. Come conseguenza è diffuso il sospetto che i costruttori di microprocessori che non producano in quantità molto grandi stiano in realtà perdendo denaro su ogni chip. Semplicemente il prezzo è oggi troppo basso perchè qualsiasi costruttore possa ricavarne un profitto su quantitativi relativamente piccoli. Pochi mesi fa la National Semiconductor ha introdotto l'8080 in contenitore plastico venduto alla Radio-Shack in quantitativi unitari per soli 17,95 $. Queste vendite sono dirette al mercato degli hobbysti. Ci si aspetta da più parti che il prezzo dei microprocessori calerà anche in quantitativi unitari a circa 10 $ entro i prossimi due anni o anche meno.

La maggior parte degli altri microprocessori ha anch'essa i propri costruttori alternativi. Il 6800, ad esempio, viene fornito in alternativa negli Stati Uniti dall'American Microsystem (A.M.I.) e in Europa dalla Sescosem Division della Thomson CSF.

Alla fine del 1976 l'Intel 8080 e il Motorola 6800 erano i due microprocessori standard più venduti. Questi due microprocessori sono anche gli unici due microprocessori monolitici standard che ci si attende verranno qualificati tra breve tempo nel programma di affidabilità militare 38510 (JAN). Essi potrebbero essere inseriti nella lista QPL per l'inizio del 1978.

Rockwell

Anche il PPS8 della Rockwell fu introdotto per competere con l'8080 e il 6800. Si tratta di un microprocessore potente anche se è realizzato con tecnologia PMOS. Alla Rockwell è stata utilizzata l'abilità autonoma nel progetto di processori derivata dalla fabbricazione di chip simili per applicazioni militari e spaziali.
Nonostante le molte caratteristiche interessanti del PPS8, questo fu introdotto sul mercato troppo tardi. La famiglia di chip di supporto al microprocessore fornita dalla Rockwell è probabilmente una delle migliori sul mercato comprendendo il primo controllore di disco su un solo chip, controllore di visualizzazione, controllore di tastiera, DMAC ecc. Probabilmente convinto che i vantaggi ovvi di questo impressionante numero di chip di supporto avrebbero garantito le vendite, alla Rockwell decise di rendere questi chip periferici incompatibili con l'8080 e il 6800. Come risultato sia questi chip, sia il PPS8 non hanno ottenuto fino ad ora un significativo successo di vendita. A questo punto è troppo tardi per attendersi che il PPS8

ottenga una rivalutazione. Esiste una nuova versione, il PPS8/2, che realizza un microcomputer completo su due chip. Esso è compatibile con il set di istruzioni del PPS8.

Signetics

Anche il 2650 della Signetics ha impiegato un lungo tempo per essere completato. La Signetics ha dovuto affrontare difficoltà finanziarie. Durante la recessione avvenuta alla metà degli anni 70, quando il dollaro raggiunse un record di svalutazione, investire nell'acquisizione di società statunitensi era una scelta particolarmente attrattiva; la Signetics fu assorbita dalla Philips (Olanda). Naturalmente il prezzo non fu l'unica motivazione. Con il finanziamento e il potere di acquisto della Philips il 2650 ottenne un'affermazione notevole. Il 2650 è un processore potente. È più lento dell'8080 e del 6800 da un punto di vista aritmetico. Tuttavia ha un'architettura originale e un set di istruzioni interessante. Il principio che ha guidato la sua realizzazione è stato che le «comunicazioni sono più impegnative dell'elaborazione». Come conseguenza di questa filosofia l'address bus è fornito di un'unità di elaborazione limitata; è possibile calcolare indirizzi molto complessi e sofisticati. Il 2650 è un microprocessore equipaggiato con uno dei più complessi set di istruzioni per l'indirizzamento della memoria. Si può realizzare l'indirizzamento indiretto, quello indicizzato, l'indicizzato indiretto ecc. In considerazione di queste possibilità di indirizzamento elaborato della memoria, il 2650 trova il suo impiego ideale ove necessitino complesse procedure di ricerca di parole in memoria. È molto adatto per il word processing e per dare accesso a catene di caratteri. Per merito dell'investimento Philips nel 2650 ci si può attendere che questo microprocessore rimanga sul mercato anche se non può competere direttamente con l'8080 o il 6800 per le applicazioni orientate ad elaborazioni aritmetiche.

National Semiconductor

La National Semiconductor ha introdotto l'SC/MP come proprio primo progetto a 8 bit. SCMP sono le iniziali di «simple-cost-effective-microprocessor». Si era inteso realizzare un semplicissimo microprocessore (che richiedesse pochi chip aggiuntivi), molto economico. Di fatto è un microprocessore semplificato ed è di fatto poco costoso. Tuttavia un basso costo per la MPU non è sufficiente a determinare vendite significative. La versione a canale P dell'SC/MP era lenta. Ora è disponibile una nuova versione in tecnologia a canale N che ha permesso di aumentare la velocità. Questo microprocessore offre una combinazione interessante di possibilità che lo rendono molto adatto per applicazioni semplici dove è desiderabile il risparmio sul componente. Esso non ha la potenza di elaborazione di un 8080, di un 6800 o di un 2650.

Nel 1976 la National ha introdotto un altro microprocessore a 8 bit; la propria versione dell'8080 (compatibile con l'Intel).

MOS Tecnology

La MOS Technology ha introdotto diversi microprocessori sul mercato. Essi fanno parte della «famiglia» 650X. Il termine può confondere: ci sono grosse differenze tra i diversi membri della famiglia. Uno dei primi membri della famiglia, il 6502, ha delle forti analogie con il 6800 e può essere considerato come un concorrente di questo. La sua organizzazione del bus è molto simile così come i suoi registri interni e il suo set di istruzioni ai corrispondenti del 6800. Il 6502 è più veloce del 6800 e i componenti speciali sviluppati dalla MOS Technology come la combinazione di memoria e di I/O, conducono a sistemi costruiti con un ridotto numero di chip, specie nei sistemi di dimensioni piccole e medie. Essi sono inoltre quotati ad un prezzo molto basso, conducendo a sistemi piccoli e medi economici con una potenzialità di elaborazione simile o maggiore di quella che si otterrebbe col 6800.

General Instruments

La General Instruments è un costruttore specializzato nei chip custom, cioé su specifica del cliente. Come risultato dei suoi progetti custom si ritrova ogni tanto con un componente che è già stato sviluppato e che può essere commercializzato autonomamente sul mercato. Periodicamente si è vista la General Instruments presentare tali microprocessori sul mercato. In genere essi non sono stati progettati fin dall'inizio per essere processori di impiego generale e non esiste alcun supporto di altri componenti. Uno di questi processori a 8 bit è l'LP8000. Esso non può essere considerato un concorrente degli altri microprocessori a 8 bit. La General Instrument ha certamente la capacità di produrre progetti avanzati. Per il momento non ha ancora deciso di utilizzare questa possibilità, almeno nel caso di microprocessori a 8 bit.

Western Digital

Il gruppo MCP 1600 della Western Digital è un caso speciale. È un progetto microprogrammato su 2 chip. La MPU richiede una memoria di sola lettura esterna con funzione di controllore che contiene il microprogramma. Il gruppo di componenti viene usato per realizzare l'LSI11 della Digital Equipment. In previsione di difficoltà produttive, la Digital Equipment costruisce ora il proprio gruppo di componenti in un apposito stabilimento sulla costa orientale. L'LSI11 è software compatibile con la linea PDP11 della Digital e sarà distribuita dalla Heathkit. È equivalente al PDP11/03.

Intersil

Anche l'Intersil 6100 è un microprocessore CMOS. Esso è un microprocessore a 12 bit. Esso è stato progettato specificamente per emulare il PDP8 della Digital Equipment, che è un processore a 12 bit. Il PDP8 è stato il minicomputer di maggior successo nel mondo. La motivazione è stata quella di introdurre una realizzazione su un unico chip della CPU del PDP8 che potesse sostituirla economicamen-

te aggiungendo la portatilità. Il 6100 fu il risultato di questo sviluppo ed è l'unico microprocessore a 12 bit sul mercato (escludendo il Giapponese TLCS12 della Toshiba sviluppato per la Ford Motor Co.). Il 6100 esegue tutte le istruzioni del PDP8. È software compatibile con questo. Un altro vantaggio è la sua realizzazione in tecnologia CMOS: un sistema completo Intersil può essere alimentato con batterie tradizionali. La compatibilità software con il PDP8 permette all'utilizzatore di programmare direttamente con le istruzioni del PDP8. Tuttavia non è possibile utilizzare i chip di ROM direttamente con questo microprocessore nello stesso modo in cui sarebbero stati usati in un microprocessore standard. Una peculiarità del PDP8 è di memorizzare l'indirizzo di ritorno delle subroutines all'inizio della subroutine. Chiaramente ciò richiede che la subroutine venga memorizzata in una RAM. Per questa ragione i programmi dell'Intersil 6100 devono venir memorizzati in una RAM. Possono anche venir memorizzati in una ROM, ma allora è necessario un «adattamento» del programma per memorizzare gli indirizzi di ritorno in una RAM separata.

Zilog

Tre tra i principali progettisti dell'8080 hanno abbandonato l'Intel creando una loro propria ditta, la Zilog a Los Altos. La Zilog è finanziata dalla Exxon, la compagnia petrolifera. Iniziando con un piccolissimo numero di persone, la Zilog ha raggiunto oggi le dimensioni di una regolare fabbrica costruttrice di semiconduttori. Il suo primo e fondamentale progetto è stato lo Z80. Lo Z80 incorpora in un unico chip: l'8080, il suo clock 8224 e il suo sistema di controllo 8228 più alcune possibilità aggiuntive. È completamente compatibile che con il software dell'8080, ha la velocità della versione più veloce dell'8080, mentre il nuovo Z80A funziona con un clock a 4 MHz contro i 3 MHz del più veloce 8080 (A—2).

Se paragonato all'8080 fornisce essenzialmente due possibilità addizionali:

1. È fornito di due banchi di registri. Ciò mette a disposizione sia un gran numero di registri interni, sia una rapida risposta ad un singolo livello di interrupt. Questi banchi di registri sono stati realizzati in modo corretto cioé duplicando l'accumulatore e la status word. Inoltre ciascun banco di registri possiede tutti i registri dell'8080. Esso ha anche un gruppo di registri indice.

2. È fornito di una facoltà di refresh per memorie dinamiche che permette il collegamento diretto di RAM dinamiche col sistema, senza la necessità di un circuito di refresh esterno.

Lo Z80 ha alcuni altri vantaggi secondari rispetto all'8080.

Due dei codici di istruzione non utilizzati sull'8080 vengono qui usati per fornire istruzioni aggiuntive, in particolare operazioni di trasferimento in blocchi. Ciò conduce a programmi più brevi ed esecuzioni più rapide. Una maggior velocità deriva

in genere più facilmente dal maggior numero di registri piuttosto che dalle istruzioni specializzate. Un punto fondamentale è che lo Z80 è compatibile a livello di ROM con l'8080. Un programma sull'8080, persino se è stato realizzato su ROM, può venir direttamente connesso nello zoccolo di un sistema con lo Z80. E funziona. Non utilizzerebbe nessuna delle caratteristiche speciali dello Z80, ma sarebbe completamente compatibile.

Infine lo Z80 è fornito di due livelli di interrupt rispetto ad uno solo dell'8080. Richiede anche un'unica alimentazione invece di tre. Lo Z80 ha creato una concorrenza significativa all'8080, specie nei casi dove è importante un aumento di prestazioni oppure dove la riduzione del numero dei chip è determinante. Tuttavia fino a poco tempo fa la Zilog non produceva i molti altri componenti della famiglia e non poteva fornire il medesimo livello di supporto offerto dalla Intel. Recentemente la Zilog ha iniziato a fornire diversi chip di supporto e la porzione di mercato dello Z80 potrebbe aumentare.

Intel

La Intel non poteva rimanere inerte mentre si delineava il pericolo presentato dalla linea di prodotto Z80. Era ovvio che con il progresso della tecnologia i chip aggiuntivi richiesti dall'8080 potessero venir integrati in un singolo chip. L'8085 è la risposta dell'Intel. L'8085 incorpora il clock 8224, il controllore di sistema 8228 e l'8080 in un unico chip. Esso funziona alla velocità dell'8080 più veloce (3MHz). L'8085 deve essere paragonato allo Z80. Esso non offre alcune delle prestazioni dello Z80. Come lo Z80, l'8085 è software compatibile con l'8080 e fornisce alcune istruzioni aggiuntive. Ma non fornisce registri addizionali o banchi multipli. Non fornisce il registro di refresh dinamico. Però mette a disposizione quattro livelli di interrupt (più un livello software), che non sono previsti nello Z80. Per ottenere questo risultato esso utilizza ancora una volta tutti i pin disponibili sul package senza mandare in uscita tutti i segnali che sarebbero necessari. Si tratta ancora di un problema di mancanza di pin. Questa volta i segnali di controllo non vengono multiplexati. Viene multiplexato il bus dei dati. Il data bus viene multiplexato tra i dati e gli indirizzi. Gli otto bit di indirizzo meno significativi appaiono sul data bus: gli otto bit superiori dell'indirizzo compaiono su un address bus di 8 bit. Ciò richiede un mantenimento (latching) e un demultiplexing esterno del data bus. Il valore significativo dell'8085 che lo redime sta nei chip speciali che la Intel ha introdotto contemporaneamente. Questo è il valore del *sistema* 8085 rispetto al *chip* 8085. Questi componenti verranno descritti con maggior dettaglio nel prossimo capitolo.

In breve essi sono combinazioni di memorie più I/O e si collegano direttamente all'8085 senza alcuna necessità di componenti addizionali: un sistema completo con l'8085 può venir assemblato esattamente con tre chip. Il mercato dell'8085 è quindi il mercato dei microprocessori potenti (la medesima potenza dell'8080) che richiedono un numero minimo di chip (meno chip che nel caso dello Z80). La limitazione del sistema è la limitazione della memoria contenuta in queste combinazio-

ni di memoria più I/O, cioé 2K parole di ROM e 256 parole di RAM. Per sistemi di dimensioni piccola o media l'8085 è probabilmente in una posizione di vantaggio. Per i sistemi potenti e complessi lo Z80 è probabilmente nella posizione dominante. È bene precisare ancora che la superiorità tecnica non dovrebbe essere l'unico criterio di scelta e gli altri criteri saranno presentati alla fine del capitolo. Un paragone tecnico tra l'8080 e l'8085 compare in Fig. 4-12.

	8080	8085
CPU	3 CHIPS: 8080+8228+CLOCK	1 CHIP
VELOCITÀ	2.µs A 1,3 µS (8080 A-1)	1,3 µS
RICHIESTE DELL'ACCESSO DI MEMORIA	300 NS	450 NS
ALIMENTAZIONE	3	1 (+5V)
SET DI ISTRUZIONI	SEE 8080	STESSO +2 ISTRUZIONI
TEMPO DI LETTURA	IDENTICO	IDENTICO
INTERRUPTS	1	5 (3 MASCHERABILI)
I/O SERIALE	0	UNO IN, UNO OUT

Fig. 4-12: 8080 ed 8085.

MICROCOMPUTER SU 1 CHIP A 8 BIT

Le caratteristiche tipiche di un microcomputer su 1 chip a 8 bit includono una ROM da 1K o 2Kx8 che contiene i programmi interni e una RAM da 256 o 512 parole che contiene i registri interni, lo stack e lo scratch pad. Le 16 linee che erano necessarie in un'architettura standard per l'address bus vengono lasciate libere per le funzioni di I/O. Un microcomputer su 1 chip a 8 bit ha perciò tre porte di I/O a 8 bit più alcune linee addizionali. In più esso include normalmente il circuito di clock come anche un contatore programmabile di eventi o timer. Poiché il processore è fornito di una memoria ROM è economico impiegarlo solo se ne prevede l'uso in quantitativi di molte migliaia di pezzi. I mercati tipici per questi microcomputer coinvolgono quantità di 10.000, 100.000 o più dispositivi. In queste quantità ci si può attendere che il costo del dispositivo sia nell'ordine di 1 $ o 2 $.

Oggi ci sono sul mercato due microcomputer su 1 chip a 8 bit. Essi sono il 3870 (l'F8 su 1 chip) prodotti dalla Fairchild e dalla Mostek e l'8048 della Intel. Presto dovrebbero venir introdotti altri microcomputers a 8 bit. Si sono sentite indiscrezioni dalla Zilog e dalla Rockwell. La versione su 2 chip dell'F8 è stato il primo microcomputer su 2 chip ad essere introdotto sul mercato. L'F8 è caratterizzato da

un'architettura piuttosto anticonvenzionale e da un set di istruzioni «inusuale». Ciò è dovuto alla sua evoluzione storica: l'F8 su due chip è derivato dalle richieste specificate da una società Tedesca per un chip custom costruito dalla General Instruments. L'F8 su 2 chip è stato introdotto successivamente come prodotto della Fairchild ed è essenzialmente simile al progetto della GI. Da allora tutte le sue funzioni sono state realizzate su un singolo chip, il 3870, dalla Mostek (il suo fornitore alternativo) ed ora dalla Fairchild. È probabile che molti più chip di questo tipo vengano venduti nel 1977-78 rispetto a qualsiasi altro tipo di microprocessore. La ragio-

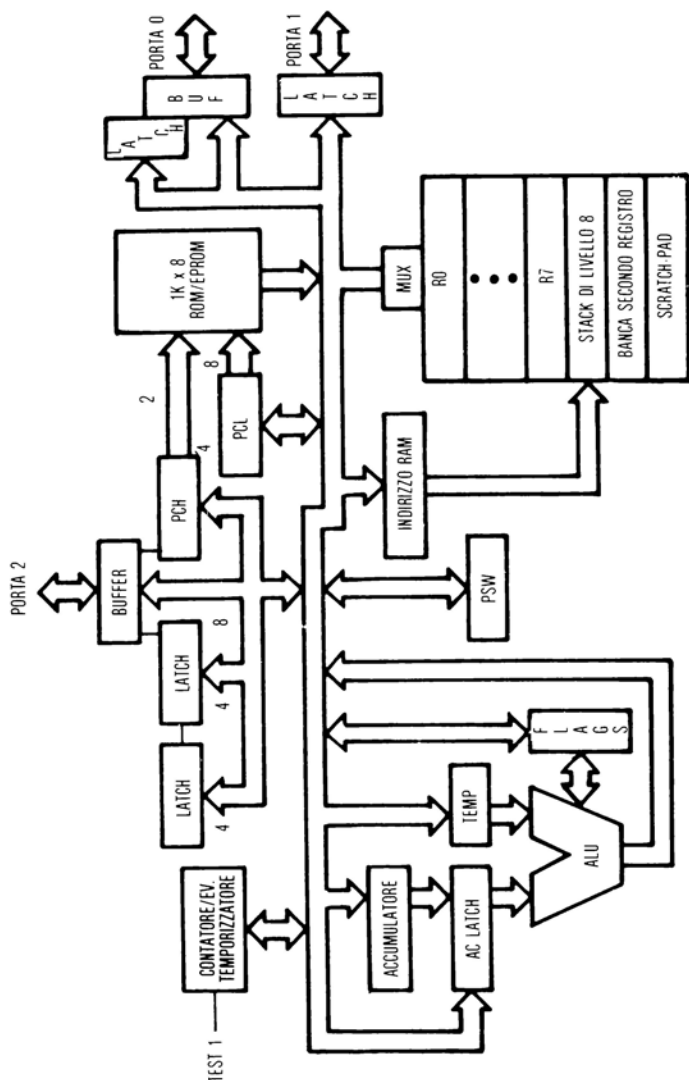



Fig. 4-13: Architettura interna dell'8048.

ne è naturalmente che il mercato obiettivo per questi chip è quello «a livello inferiore»: applicazioni industriali e consumer che implicano decine di migliaia di unità.

Un fatto interessante è che l'alto volume di vendite non implica una più ampia accettazione del chip stesso. Questi chip vengono infatti venduti ad una base molto ristretta di acquirenti.

L'architettura e il set di istruzioni dei chip diventa familiare di conseguenza solo ad un piccolo numero di utilizzatori. Questo tipo di limitazione di mercato non conduce al medesimo risultato di «formazione di abitudine» come nel caso dei chip di microprocessore standard. L'F8 è disponibile oggi per 10 $ in quantità minime di 100

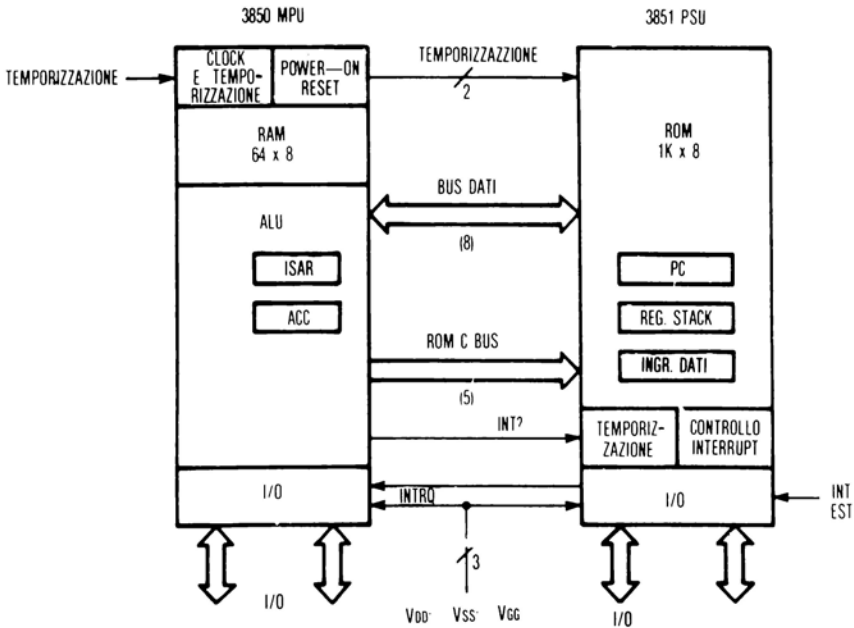



Fig. 4-14: F-8 su 2 chip.

pezzi. Poichè l'F8 non è stato progettato sin dall'inizio per essere un microcomputer su 1 chip, il suo set di istruzioni è meno facile da usare e da capire della maggior parte dei set di istruzioni tradizionali. La sua programmazione in un primo tempo è significativamente più difficile. Tuttavia il costo iniziale di programmazione non è un criterio significativo nella valutazione di un microcomputer di questo tipo. Il costo della programmazione, diviso per il numero di unità vendute è minimo. Il criterio fondamentale è quello di accertarsi se le prestazioni del dispositivo sono sufficienti e il suo prezzo in quantitativi.

Il principale concorrente esistente dell'F8 è l'8048 dell'Intel. L'architettura dell'8048 è derivata dall'8080, ma non è compatibile con esso. L'8048 ha un set di istruzioni pulito e interessante che non è il set di istruzioni dell'8080. In particolare esso finalmente permette la possibilità di testare qualsiasi bit nell'accumulatore (e di saltare direttamente ad un indirizzo in funzione di quel bit). Il principale svantaggio sul mercato incontrato dall'8048 deriva dall'essere stato introdotto significativamente più tardi della versione su un solo chip dell'F8. Fino ad ora la storia ha insegnato che esiste una tendenza a favore dei prodotti introdotti per primi sul mercato. Tuttavia è stato evidenziato che il mercato per questi prodotti è molto piccolo rispetto al numero di persone. Perciò l'impatto dell'«essere i primi sul mercato» potrebbe non essere tanto significativo come lo è stato nel caso dei microprocessori standard.

Il mercato dirà entro breve tempo se le vendite dell'8048 saranno all'altezza delle vendite dell'F8 oppure no. Le prestazioni dell'F8, quando questo venga utilizzato da un programmatore esperto, si avvicinano a quelle dell'8048 sotto molti aspetti.

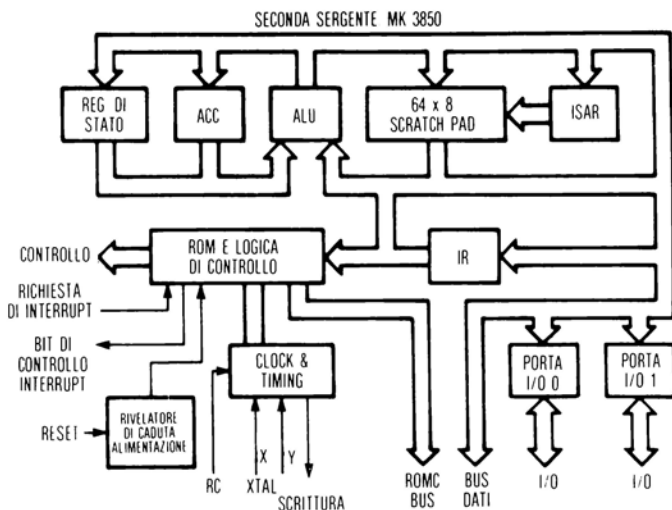



Fig. 4-15: MPU 3850 F8 della Fairchild.

Una delle peculiarità dell'8048 è la disponibilità di una versione con EPROM (l'8748). L'8748 è un 8048 fornito di EPROM cancellabile con raggi UV, al posto della ROM standard. Essa può essere cancellata con luce ultravioletta e riprogrammata. Questo è un vantaggio durante la fase di sviluppo. Questo probabilmente non è un vantaggio molto significativo perchè gli utilizzatori di microcomputer su 1 chip, prevedendo la produzione su larga scala di migliaia di dispositivi normalmen-

te possono investire in qualunque sistema di sviluppo si dimostri necessario per programmare il componente. Tuttavia questa è certamente una comodità che renderà accessibile il chip al piccolo utilizzatore.

I microcomputer su 1 chip, siano essi l'F8, l'Intel 8048 o i loro concorrenti che verranno annunciati al più presto, avranno un impatto determinante sul mercato (a ragione dei quantitativi implicati) così come sui prodotti industriali e consumer. Essi stanno introducendo la capacità di una «intelligenza» limitata a prezzi che si avvicinano a 1$ in grandi quantitativi. Ci si può attendere che la maggior parte dei prodotti consumer che possono trovare vantaggio da un qualsiasi grado di automazione verranno forniti in futuro con tali dispositivi.

Fig. 4-16: PSU F8 (3851).

MICROPROCESSORI A 16 BIT

A tutt'oggi sono stati introdotti sul mercato di fatto cinque microprocessori a 16 bit. Eccettuato eventualmente per uno di essi, non sono stati intesi per competere sul mercato del «microprocessore standard». Non possono farlo. La limitazione fondamentale dei microprocessori a 16 bit è già stata puntualizzata: è la limitazione sul numero dei pin (al massimo 40 o 42). È stato dimostrato che entro i 40 pin non è possibile realizzare un address bus a 16 bit, un data bus a 16 bit e un ragionevole control bus. Uno dei bus deve venir multiplexato. Ciò conduce alla necessità di latch esterni per gli indirizzi, circuiti di demultiplexing ed in un significativo rallentamento del sistema. Una volta che i dati o le istruzioni di 16 bit sono entrate nel microprocessore la velocità di elaborazione risulta significativamente più alta che nel caso del microprocessore a 8 bit. Il problema sta nel far entrare ed uscire questi segnali.

NOTA i microp INTEL 8086 e ZILOG Z8000 sono stati annunciati ma non sono ancora disponibili sul mercato

DATA GENERAL MN601

FAIRCHILD NOVA 1200

GENERAL INSTRUMENT CP 1600

NATIONAL PACE

TEXAS INSTRUMENT TMS 9900

Fig. 4-17: Microprocessori a 16 bit.

Inoltre, considerando la complessità di realizzazione di una tale unità a 16 bit, la velocità media di esecuzione è spesso più bassa di quella di un microprocessore standard a 8 bit.

Di conseguenza la maggior parte dei microprocessori a 16 bit oggi sul mercato non possono competere con i microprocessori a 8 bit. Essi sono indirizzati a un mercato differente. Sono essenzialmente indirizzati al mercato di *sostituzione al minicomputer.*

La **National Semiconductor** è stata la prima società ad aver introdotto un progetto a 16 bit. Ha introdotto il PACE come sostituto della CPU del «minicomputer» IMP16 di propria costruzione. L'IMP16 era un sistema scaturito dalla famiglia di bit slice GPCP della stessa National Semiconductor, che era diventato troppo lento e richiedeva troppi componenti. Il set di istruzioni del PACE è compatibile

con il set di istruzioni dell'IMP16. Sfortunatamente il PACE è realizzato con tecnologia PMOS e risulta comparativamente lento. Poiché l'IMP16 non è mai diventato un minicomputer molto diffuso, anche il PACE non ha ottenuto un'ampia diffusione.

La **General Instruments** ha introdotto il CP1600, un processore a 16 bit veloce in NMOS. Un fatto essenziale risiede nel fatto che la ITT ha annunciato che si inserirà come commercializzatrice e come fornitore alternativo per il prodotto. Ci si può aspettare che questo microprocessore a 16 bit venga in futuro usato in particolare su diversi progetti interni della ITT di tipo minicomputer.

La **Data General** ha introdotto l'MN601 per poter competere sul mercato inferiore delle applicazioni meno impegnative. L'MN601 realizza su un chip la maggior parte delle funzioni della CPU del NOVA/3. (Questo è il minicomputer NOVA più lento della famiglia). Un problema importante con il quale recentemente i costruttori di minicomputer si sono dovuti scontrare è la realizzazione dei loro minicomputers più lenti da parte di altre ditte che utilizzano dispositivi a microprocessore. Qualsiasi realizzazione LSI utilizza un minor numero di componenti e conduce ad un costo molto più basso. Per rimanere concorrenziali in questo settore di mercato i costruttori di minicomputer oggi devono produrre i loro specifici progetti LSI per ridurre il numero di componenti e i costi. Ci sono tutte le probabilità che la Data General non intenda supportare l'MN601 come dispositivo microprocessore, ma che intenda commercializzarlo come scheda della linea NOVA. Il vantaggio fondamentale di questo sistema per l'utilizzatore è naturalmente il fatto che tutto il

	DATA GENERAL MN 601	GEN. INST. CP 1600	NS PACE	TEXAS TMS 9900	FAIRCHILD 9440
TIPO	STANDARD	STANDARD	STANDARD	STANDARD	STANDARD
TECNOLOGIA	NMOS	NMOS	PMOS	NMOS	I3L
NUMERO ISTRUZIONI	NOVA 3 INSTRUCTIONS	87	50	69	NOVA 1200 SET
TEMPO DI CICLO (μS)	2,4 a 10	2,4	8 a 10	1,5	1 a 2,5
INDIRIZZAMENTO DIRETTO (BIT)	16	16	16	16	16
REGISTRI	4	7	4	RAM (16)	4
STACK (LIVELLI)	HARD	SOFT	HARD (10)	--	HARD
INTERRUPTS	1	2	6	16	1
CLOCK SUL CHIP	si	--	--	--	--
ROM SUL CHIP (BYTES)	--	--	--	--	--
RAM SUL CHIP (PAROLE)	--	--	--	--	--
TEMPORIZZATORI SUL CHIP	si	--	--	--	--
FFR	si	--	--	--	--
ALTRI	--	--	--	--	--
LINEE I/O	--	--	--	--	--
CONTENITORE (PINS)	40	40	40	64	40
ALIMENTAZIONE (V)	-4.25, +5, +10, +14	+12, +5, -3	+5, -12	-5, +5, +12	+5, +1
SECONDA SORGENTE	--	ITT	--	--	--
NOTE	EMULATORE NOVA 3	--	COMPATIBILE DIP 16	MOLT/DIV DI INTERI	EMULATORE NOVA 1200

Fig. 4-18: Tabella comparativa dei microprocessori a 16 bit.

software sviluppato per il medesimo è compatibile con il NOVA e può essere direttamente sviluppato su di un minicomputer NOVA.

La **Fairchild** ha annunciato il microprocessore a 16 bit 9440, che realizza la maggior parte delle funzioni della CPU di un NOVA 1200 su un unico chip. Sarebbe quindi in diretta concorrenza con i prodotti Data General. Tuttavia in considerazione delle complessità del progetto, sembra che la Fairchild stia tuttora cercando di risolvere alcuni problemi di produzione e che il costo del dispositivo debba essere elevato. Se e quando verrà distribuito in grandi quantitativi, questo microprocessore potrà essere di interesse particolare per tutti gli utilizzatori del software Data General.

La **Texas Instruments** ha introdotto uno dei più veloci processori monolitici oggi sul mercato: si tratta del TMS 9900. In particolare esso è fornito di operazioni hardware di moltiplicazione e divisione. Tuttavia la sua velocità ha un prezzo: il dispositivo ha 64 pin. È un processore potente non più limitato dalla restrizioni dei 40 pin e può comunicare con il mondo esterno mediante un address bus a 16 pin nonchè con un data bus a 16 pin. Esso è ben supportato dalla Texas Instruments che ha introdotto diversi dispositivi di supporto compatibili con il 9900 (come anche l'8080 per il quale è fornitore alternativo). Il 9900 viene utilizzato internamente dalla Texas Instruments per la propria linea di minicomputer/microcomputer 990. Uno dei vantaggi del 9900 sta nel fatto che si presenta come un potente equivalente su un chip di una CPU di un minicomputer. Poiché la Texas Instruments ha usato questo chip come CPU del proprio sistema 990; qualsiasi utilizzatore del 9900 realizzerà di fatto un processore simile ad un minicomputer con il medesimo set di istruzioni del 990. Il 9900 può essere interessante per gli utilizzatori che ricercano forza bruta in termini di capacità di elaborazione su un singolo chip. Uno degli aspetti interessanti del 990 sta nel fatto che alla sua introduzione è seguita una serie di altri dispositi compatibili come codice con il 9900. In particolare il 9940, il nuovo microcomputer su 1 chip della TI, ha un set di istruzioni compatibile con il 9900. Se quest'ultimo componente dovesse stabilire una testa di ponte come successo commerciale, ciò affermerebbe come conseguenza il set di istruzioni e di conseguenza le vendite dello stesso 9900. Altri dispositivi simili al 9900 sono disponibili con formato di 8 bit.

MICROCOMPUTER SU 1 CHIP A 16 BIT

Fino ad ora è stato annunciato ufficialmente un unico microcomputer su un unico chip a 16 bit. Si tratta del 9940 della Texas Instruments, con un set di istruzioni compatibile con il 9900. Tuttavia al momento della stesura di questo testo (n.d.t. quello originale in inglese, naturalmente) non è ancora disponibile. È opinione diffusa che un simile componente verrà introdotto da altri costruttori leader entro breve tempo. È importante precisare che fintanto che tali dispositivi non sono disponibili in grandi quantitativi, dovrebbero venir considerati come del tutto non disponibili. La storia ha dimostrato che molti dispositivi che erano stati annunciati o che si spe-

rava venissero distribuiti, in realtà non sono mai entrati in produzione di serie. Per questa ragione si deve fare molta attenzione. Tuttavia i microcomputer su 1 chip a 16 bit rappresentano la soluzione tecnica al problema della limitazione sul numero dei pin che è stato spiegato. Un microcomputer completo a 16 bit integra la memoria entro il medesimo chip. Non richiede più un address bus di 16 bit e può usare i pin disponibili per le funzioni di input e di output. Questa è realmente la soluzione del futuro. Inoltre se questo minicomputer fosse compatibile come set di istruzioni con un minicomputer già esistente, potrebbe utilizzare tutto il suo software e risolvere nello stesso tempo il problema della programmazione.

La disponibilità di 16 bit entro il chip conduce ad un aumento di velocità d'esecuzione per le operazioni logiche ed aritmetiche. In più permetterebbe l'uso di potenti codici operativi di 16 bit simili a quelli dei minicomputer, invece di codici operativi di 8 bit. Ciò condurrebbe, d'altra parte, ad istruzioni molto più complesse. In breve non appena i microcomputer a 16 bit diverranno realmente disponibili in grandi quantità, si può prevedere che essi si impadroniranno di una porzione significativa del mercato precedentemente tenuto dai microprocessori «più piccoli». Essi rappresentano il progetto di microcomputer standard del futuro.

Si può immaginare che in futuro verranno introdotti modelli con più ampie dimensioni di parola. Tuttavia ciò è del tutto a livello speculativo a questo punto. Sebbene le densità vengano costantemente aumentate, il costo di questo aumento e le difficoltà sono anch'essi aumentati costantemente. Quindi si può prevedere che i microcomputer a 16 bit su un unico chip saranno disponibili entro pochi anni o anche meno. Non è possibile fare una previsione ragionevole riguardo altri dispositivi più complessi.

- NS IMP
- INTEL 3000
- MMI 5701
- AMD 2901
- MOTOROLA XC 10800
- TEXAS SBP 0400 ——→ 0401

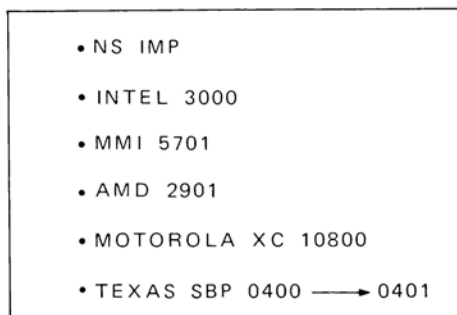



Fig. 4-19: Bit - slices.

PROCESSORI BIT SLICE

Un dispositivo *bit slice* non dovrebbe essere chiamato propriamente *microprocessore.* Il termine «microprocessore» è applicato spesso ai dispositivi suddivisi per porzioni di bit perchè sono componenti LSI che realizzano alcune delle funzioni di

una CPU. Essi non sono CPU complete; per questa ragione ci asterremo dall'etichettarli *microprocessori*. Un bit-slice è una slice (fetta) di un'unità logica aritmetica completa dei propri multiplexer e delle linee dei dati. Un bit-slice è una linea completa di dati simile alle corrispondenti linee che si trovano nella CPU di un processore tradizionale. Esso include: i registri, la corrispondente slice di ALU, i flag, i multiplexer, i bus. Esso esclude specificamente la sezione di controllo. La sezione di controllo di un sistema bit slice deve essere realizzata con dispositivi esterni ed è generalmente microprogrammata. Tuttavia essa richiede un numero significativo di dispositivi addizionali che rendono un progetto bit-slice piuttosto complesso.

	INTEL 3000	MMI 5701	AMD 2901	MOTOROLA XC 10800	TEXAS SBP0400
NUMERO DI BIT	2	4	4	4	4
TECNOLOGIA	SHOTTKY	SHOTTKY	SHOTTKY	MECL	I2L
ISTRUZIONI	MINIME	36	36	COMPLESSE	16
TEMPO DI CICLO (MS)	100	200	95, 125	55	1000
SUL CHIP - REGISTRI	12	16	16	---	7
-REG DI MEMORIA	SI	---	---	---	---
-ACC ESTERNO	---	SI	SI	---	---
-STACK	---	---	---	---	---
-STATO	SI	SI	SI	SI	SI
PACKAGING	28	40	40	48	40
SECONDA SORGENTE	SIGNETICS	---	RAYTHEON, MOTOROLA THOMSON	---	---
NOTE	---	---	STESSA ARCHITETTURA COME 5701	---	128mW

Fig. 4-20: Tabella comparativa dei bit-slices.

La misura ottimale corrente per i bit-slice è di quattro bit. I bit-slice sono diventati la pietra miliare per costruire oggi i più potenti processori, riducendo contemporaneamente in modo significativo la lista dei componenti necessari per la CPU. Inoltre essi hanno aperto la strada a nuovi progetti che includono progetti multi-bit-slice.

L'introduzione dei bit-slice ha preso il suo avvio dalla ricerca di una maggiore velocità. Per realizzare la più alta velocità di commutazione possibile i progettisti di

componenti avevano scelto le più veloci tecnologie disponibili come la bipolare e, in particolare, la low-power Shottky (LPS TTL). Sfortunatamente la tecnologia bipolare ha un consumo di potenza significativo così come la relativa dissipazione di potenza e richiede ampie aree di silicio per essere realizzata su un chip. Come risultato l'integrazione del componente è bassa e non sarà possibile per il prossimo futuro integrare un microprocessore bipolare completo su un unico chip. I progettisti hanno dovuto limitare le funzioni che stavano realizzando. Si può facilmente dimostrare che solo l'integrazione verticale, come alternativa al progetto orizzontale, ha un futuro nel campo dei componenti LSI. Le densità LSI semplicemente non pos-

	NS IMP
NUMERO DI BIT	4
TECNOLOGIA	PMOS
ISTRUZIONI	4 + SHIFT
TEMPO DI CICLO (MS)	9200
SUL CHIP - REGISTRI	7
-REG DI MEMORIA	---
-ACC ESTERNO	---
-STACK	SI (16)
-STATO	SI
ALIMENTAZIONE	-12, +5
PACKAGING	24, 40

Fig. 4-21: Tabella comparativa dei bit-slices (segue).

sono essere ottenute con un progetto orizzontale. Come conseguenza è stata introdotta sul mercato una slice altamente integrata verticalmente attraverso una CPU completa. Essa è il dispositivo standard a slice di 4 bit corrente. Diverse ditte hanno introdotto *slice* di ALU. Bisogna precisare che essi non erano *dispositivi bit slice*. Slice di ALU di 4 bit sono state usate già da diverso tempo per realizzare ALU a 8 o 16 bit. Tuttavia esse non erano nient'altro che quello, fette di ALU. Un bit-slice incorpora tutti i multiplexer, i bus, i registri, e le flag. In breve se ponessimo in parallelo quattro bit-slice di 4 bit ne deriverebbe una ALU completa a 16 bit con i

propri registri e con almeno tre bus, più i flag usuali.
Vedremo ora una breve storia dei dispositivi bit-slice.

La **National Semiconductor** è stata la prima società ad introdurre un dispositivo bit-slice: fu il gruppo GP/CP, in tecnologia PMOS, che venne usato per realizzare la serie di sistemi tipo minicomputer IMP. Si può notare la forte influenza del minicomputer Nova della Data General. Il risultato fu un minicomputer a 16 bit IMP16 realizzato con quattro slice più gli elementi di controllo (le CROM). Questi progetti realizzavano un minicomputer completo con un piccolo numero di chip. Sfortunatamente erano molto lenti. Tuttavia sono stati usati come ibridi per applicazioni militari che società come la Teledyne (TDY 52B). Il GP/CP può essere considerato il nonno dei bit-slice. Oggi è obsoleto.

La **Intel** fu la prima società ad introdurre un dispositivo bit-slice bipolare veloce, il 3000. Il 3000 fu un vero bit-slice, veloce (100 μsec). Il 3000 per il fatto di essere stato il primo progetto bipolare di un vero bit-slice fu un notevole salto nel buio. Oggigiorno si possono notare le sue inadeguatezze o svantaggi: opera solamente su due bit e richiede perciò più componenti di uno slice a 4 bit. Anche il suo set di istruzioni è limitato. Tuttavia queste non sono limitazioni significative considerando la grande versatilità di questo chip, che è fornito di un gran numero di bus. Una caratteristica del 3000 è contemporaneamente un vantaggio significativo e il suo svantaggio più evidente: esso è *microprogrammabile orizzontalmente.* Una microistruzione specifica simultaneamente eventi entro diversi campi. Di conseguenza è possibile eseguire molte operazioni simultaneamente ed ottenere una grande velocità purchè il microprogrammatore sappia come trarre vantaggio da ciò. Sfortunatamente molti microprogrammatori vengono semplicemente scoraggiati dall'apparente complessità che ne deriva e considerano questa prestazione uno svantaggio.

Il 3000 è particolarmente adatto a controlli complessi in tempo reale di processi sofisticati. È stato usato in controllori di disco, come anche in numerose applicazioni avioniche (signal processing). È utilizzato da costruttori di calcolatori per uso militare come la Hughes con l'AN/UYK30. Uno degli svantaggi nella programmazione del 3000 sta nel fatto che ciascuna microistruzione contiene l'indirizzo della successiva.

A causa della limitazione sul numero dei pin, il campo degli indirizzi è limitato a sette bit. Ciò determina convenzioni di jump complesse. In pratica questo significa che le istruzioni devono essere depositate nella memoria da un programma speciale che pone le microistruzioni in modo tale che sia possibile saltare da ciascuna di esse alla successiva. Il problema a livello di debugging sta nel fatto che, una volta che sia stato individuato un errore, generalmente si deve rimappare il programma completo in memoria. Ciò è accettabile per applicazioni complesse. Il medesimo fatto viene normalmente giudicato un ostacolo nelle applicazioni semplici.

La **Monolithic Memories** (MMI) ha introdotto il 5701/6701 come precursore della versione standard a 4 bit usata oggi. Il progetto MMI fu ben presto seguito

dall'AMD 2901 che verrà descritto nel prossimo paragrafo. In breve l'MMI 5701 fu probabilmente indirizzato alla realizzazione di una emulazione del Nova 800 o 1200 della Data General. La MMI assemblò delle schede utilizzanti un progetto bit slice con il 6701 che realizzava le funzioni di una scheda del Nova Data General con molto meno componenti del Data General ed un aumento di velocità del 10%. La introduzione del 2901 da parte della AMD ha reso superati i 5701/6701 offrendo funzioni identiche ed una più alta velocità.

La **Advanced Micro Devices** (AMD) ha introdotto il 2901 che è diventato oggi il bit slice a 4 bit standard. L'architettura del 2901 è essenzialmente identica a quella del 5701 con qualche miglioramento. Le sue prestazioni sono migliorate sensibilmente (105 nsec invece di 190 nsec ed una velocità ancora più elevata nella nuova versione denominata 2901A). I miglioramenti del 2901 rispetto al 5701 consistono per esempio in un bit in più nel campo delle microistruzioni che determinano un raddoppio delle microistruzioni e un collegamento esterno sul bus interno.

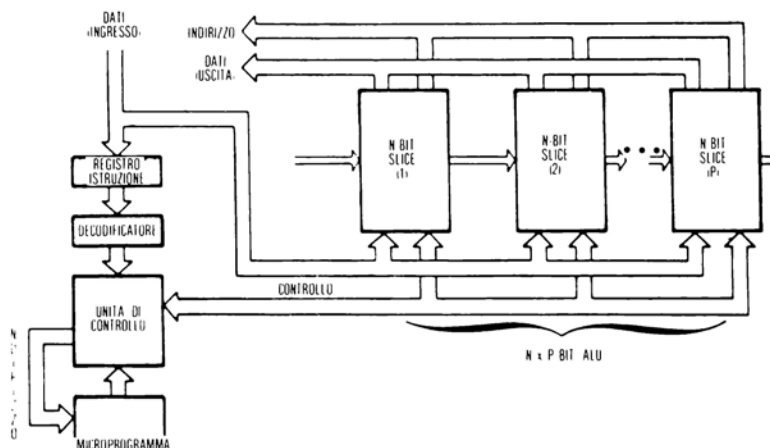



Fig. 4-22: «Sistemi» bit - slice.

Una ALU a 16 bit può venir realizzata con questo dispositivo mettendo in cascata (ponendo in parallelo) quattro slice. Per una maggior efficienza nelle operazioni aritmetiche in genere verrà aggiunta un'unità di look ahead sul riporto (carry) assieme ad alcuni componenti discreti. La reale complessità e limitazione sta nella *sezione di controllo* che include la logica di controllo, il contatore di loop, i multiplexer dei bus. Ad esempio l'architettura di un dispositivo bit slice compare in Fig. 4-22. La complessità del controllo esterno richiesto per costruire un sistema diminuisce con l'introduzione di nuovi componenti di controllo. Il successo dei dispositivi bit-slice nella costruzione di CPU più ampie è stato tale che sono stati resi disponibili componenti nuovi per ridurre significativamente il numero totale dei com-

ponenti per la logica esterna. I bit-slice possono essere dichiarati come gli attuali mezzi standard di progettazione per le CPU di oggi. Ad esempio una CPU realizzata con un dispositivo bit-slice standard quale il 2901 raggiunge una velocità da 220 a 300 nsec per 16,32 o più bit. L'AMD 2901 viene oggi fornito in alternativa da diversi costruttori includendo Motorola, Signetics, Thomson-CSF in Europa ed altri ancora. Esso è stato utilizzato nel progetto di diversi minicomputer correnti ed altrettanto di processori militari (dall'ATAC16 all'LSI 11M).

Alcuni altri dispositivi bit-slice sono stati introdotti sul mercato con minore successo e verranno menzionati poichè alcune delle loro caratteristiche li rende preziosi per applicazioni specifiche.

La **Motorola** ha introdotto un componente ECL, il 10800. Esso raggiunge velocità altissime, con un tempo di ciclo di 55 nsec. Sfortunatamente richiede un chip di registro esterno, che aumenta il totale dei componenti e riduce la velocità effettiva del sistema. Se questa tecnologia dovesse raggiungere un fattore di integrazione più alto, potrebbe diventare la versione più efficiente dei dispositivi bit-slice per applicazioni molto veloci.

La **Texas Instruments** ha introdotto lo 0400, seguito dallo 0401, il primo dispositivo realizzato con tecnologia I2L. La I2L (Integrated Injection Logic) è una terminologia bipolare che in breve è caratterizzata da una velocità tipica dei bipolari e da bassissimo consumo di potenza. Il bassissimo consumo di potenza rende possibile considerare la realizzazione di un sistema completo in I2L che potrebbe essere *portatile.* La I2L è una tecnologia che è stata derivata direttamente dal mercato dei calcolatori tascabili e degli orologi digitali. Essa fu sviluppata specificamente in previsione della ridotta disponibilità di potenza per questi dispositivi. Sfortunatamente i dispositivi I2L non hanno ancora raggiunto le caratteristiche di velocità dei bipolari. Ad esempio lo 0400 raggiunge solamente 1 μsec. rispetto ai 100 nsec. circa per l'AMD2901. Gli esperimenti di laboratorio hanno indicato che si possono ottenere velocità molto più elevate. Il giorno in cui la I2L avrà realmente raggiunto queste velocità più alte la sua adeguatezza per un'alta integrazione dei componenti, l'alta velocità ed il basso consumo di potenza la renderanno una candidata molto interessante per la realizzazione di calcolatori su larga scala alimentati a batteria. Tuttavia questo risultato non sembra ottenibile nel futuro immediato.

La **Fairchild** ha introdotto diversi dispositivi appartenenti alla famiglia Macrologic 9440 che non costituiscono dei bit-slice completi, pur essendo più che delle semplici slice di ALU. Essi sono in una posizione intermedia e potrebbero essere presi in considerazione come standard di progetto per calcolatori.

In pratica, al momento, la soluzione 2901 è divenuta la soluzione standard di progetto per gli slice a 4 bit sul mercato. Non appena saranno migliorate delle nuove tecnologie questa posizione di monopolio potrebbe essere sostituita da un'altro dispositivo.

SOMMARIO DEI PARAGONI

È stata presentata una valutazione comparativa dei microprocessori, tipo per tipo, per ogni costruzione. Si spera che questa discussione possa aver chiarito la giungla di prodotti disponibili oggi. Non è stato ancora risolto un interrogativo di base: quale microprocessore dovrei comperare? Nel prossimo paragrafo verranno considerati i criteri possibili per la scelta di un microprocessore.

SCELTA DEL MICROPROCESSORE

Il primo criterio da considerare nella scelta di un microprocessore è ovvio: può svolgere il lavoro?

– **1° Criterio: Potenzialità sufficiente**

Naturalmente la prima preoccupazione dell'utilizzatore dovrebbe essere quella di scegliere un componente che gli fornirà risorse adeguate per svolgere il lavoro previsto.

Si può stabilire ora una semplice classificazione: se il lavoro da svolgere dovesse richiedere una «super velocità», cioè l'esecuzione di istruzioni complesse entro il tempo di 1 μsec, nessun microprocessore monolitico lo potrebbe svolgere. Sarebbe necessario considerare una configurazione bit slice o l'uso di un minicomputer o di un altro tipo di processore. Se le istruzioni sono dello stesso ordine di complessità di quelle correntemente disponibili sui microprocessori e richiedono prestazioni più modeste di quelle che si ottengono con questi, si può considerare qualsiasi tipo di microprocessore. I microprocessori a 4 bit possono essere spesso eliminati sul piano tecnico se la loro velocità di elaborazione non è sufficiente. Oltre a questo punto la scelta diventa più difficile. Se l'elaborazione da svolgere (task) è abbastanza specifica, è possibile far uso di programmi di verifica (benchmark programs). I *benchmark programs* sono programmi «tipici» scritti dall'utilizzatore usati per verificare la velocità operativa del processore considerato. Ad esempio un programma di block-transfer o un programma di serializzazione sono tipici benchmark. Sfortunatamente nella maggior parte dei casi i programmi sono degli aggregati di funzioni e non è facile stabilire un singolo benchmark. Si dovrà allora usare «giudizio» oppure, se non disponibili più risorse e tempo, la simulazione. Si è scoperto che in pratica la gran maggioranza di tutti i lavori svolti dai microprocessori possono essenzialmente essere svolti *virtualmente da qualsiasi microprocessore standard a 8 bit o a 16 bit* o, perfino, dai microprocessori a 4 bit (sebbene ne derivino in tal caso più difficoltà ed un maggior numero di istruzioni per il programma).

Dovranno essere considerati perciò altri criteri.

Il secondo criterio fondamentale è il numero di unità da produrre:

– **2° Criterio: Numero di unità da produrre**

La domanda fondamentale è: si prevede di produrre un piccolo numero di unità

(da 100 a poche migliaia) o dei grandi quantitativi (da 10.000 a 100.000)? Il diagramma delle scelte appare in Fig. 4-23. Consideriamo il caso di una produzione su larga scala. Essa implica un costo dell'hardware minimo. Su produzioni molto ampie può essere giustificato perfino il costo di un chip custom. Esso offre il vantaggio addizionale del progetto esclusivo. Altrimenti la soluzione è quella di un microcom-

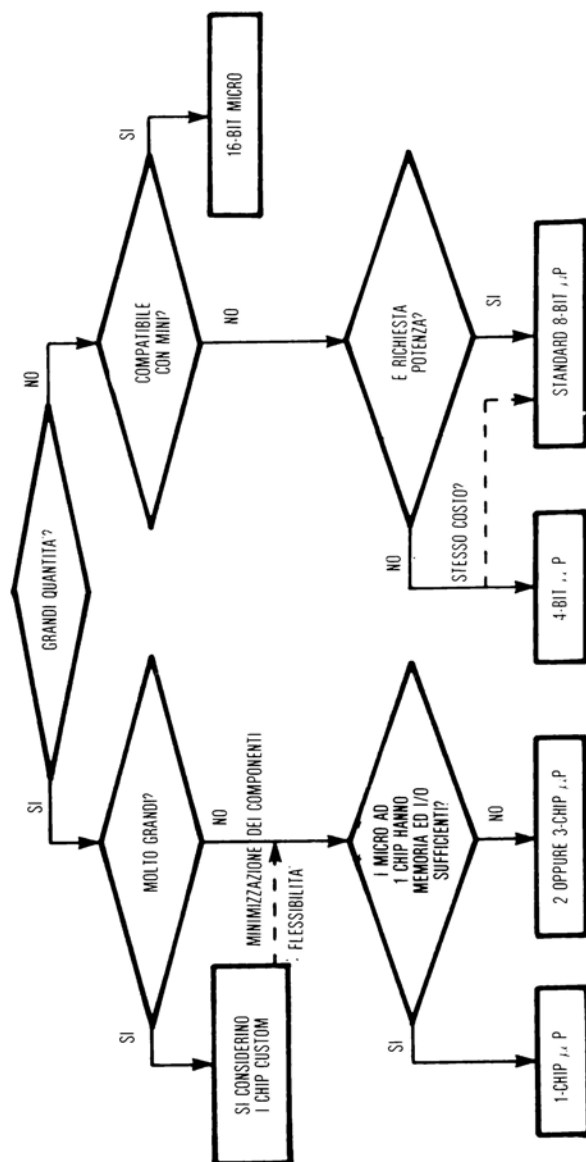



Fig. 4-23: Scelta di un tipo di microprocessore.

puter su 1 chip purchè la dimensione della sua ROM sia sufficiente. Se non lo è si dovrà considerare un progetto a 2 o 3 chip purchè esso renda disponibile abbastanza memoria. Se non è così si dovranno considerare altre soluzioni di progetto.

Quando i volumi di produzione non sono così notevoli (meno di 10.000 unità) si possono considerare due aspetti: 1. la compatibilità con un mini richiede un microprocessore a 16 bit. 2. La potenza di elaborazione va a favore di un progetto a 8 bit piuttosto che di uno a 4 bit. Se il fattore costo va minimizzato con tutti i mezzi, potrebbero essere presi in considerazione i microprocessori a 4 bit. Tuttavia il dominio dei volumi dal piccolo al medio è dominato dal microprocessore «standard» a 8 bit.

La selezione non è ancora completa e deve essere raffinata. Consideriamo l'altro criterio.

— **3° Criterio: La disponibilità.**

Il successivo criterio essenziale per valutare un sistema consiste nello stabilire se esso esiste realmente. In altre parole un sistema richiederà diversi altri chip oltre al microprocessore. Esso richiederà dei chip di supporto: chip di memoria, chip di I/O, chip di interfaccia e possibilmente dei controllori per il dispositivo. Non è ragionevole considerare la realizzazione di un processore completo su una singola scheda e dover poi realizzare le interfacce richieste di input-output su molte più schede di logica. L'intero sistema dovrebbe essere in LSI e includere il minor numero possibile di componenti. Soltanto pochi sistemi si qualificano sotto questo punto di vista. Non tutti i costruttori hanno affrontato il costo di un tale sforzo di progettazione. Come risultato, una volta che le funzioni complete del sistema sono note, diventa molto più facile considerare quali microprocessori sono disponibili, assieme ai componenti LSI richiesti, per facilitare un progetto completo. Si possono considerare ancora altri criteri.

— **4° Criterio: Software**

La facilità di programmazione può rappresentare il maggior ostacolo nel primo sforzo di sviluppo. Assiemare un sistema microprocessore è semplice, come dimostrerà il capitolo seguente. Il problema reale durante la prima realizzazione, sta nell'attività di programmazione. Per facilitare questa programmazione sono disponibili diverse risorse che verranno descritte al Capitolo 9. Se un utilizzatore richiede che sia disponibile un linguaggio ad alto livello per il microprocessore specifico, perchè ciò gli renderà la programmazione molto più facile, questo fatto dovrebbe essere considerato come un importante criterio di selezione. D'altra parte se l'utilizzatore è abbastanza sofisticato ed esperto da programmare in linguaggio assemblatore, probabilmente egli potrà programmare di fatto qualsiasi microprocessore oggi sul mercato.

— **5° Criterio: Criteri Speciali**

In previsione di richieste non standard dei clienti si dovrebbero poter considerare

diversi altri criteri speciali. Se è essenziale il basso consumo di potenza, allora sarà imperativo un progetto in CMOS. In seguito potrebbe venir considerato un progetto in I2L, allorquando diverrà disponibile un microprocessore completo con questa tecnologia.

Per i progetti militari potrebbe essere necessario considerare dei microprocessori che abbiano le caratteristiche di resistenza all'ambiente richieste. Oggi la maggior parte dei microprocessori sono disponibili in una versione «M» che potrà sopportare la gamma estesa di temperatura assieme ad alcune altre specifiche militari. Tuttavia al momento in cui è stato scritto questo libro nessun microprocessore è realmente qualificato MIL secondo le specifiche della 38510 (programma JAN). Nel prossimo futuro dovrebbero venir qualificati tre microprocessori: essi sono l'Intel 8080, il Motorola 6800 e l'AMD 2901.

SOMMARIO

In questo capitolo sono stati presentati, comparati e valutati tutti i tipi di microprocessori. La scelta del microprocessore più adatto è più semplice di quanto potrebbe apparire a prima vista. Ogniqualvolta viene individuato un criterio predominante come, ad esempio, la ricerca di un'alta velocità, del basso costo o di un basso consumo di potenza, la scelta viene ad essere molto ristretta e perciò semplice. Quando non esiste alcun criterio predominante la scelta è, come al solito, un compromesso. I criteri presentati dovrebbero permettere di restringere il numero di microprocessori da prendere in considerazione. Il punto di vista fondamentale che viene suggerito è di prendere in considerazione il parametro più significativo (includendo anche il costo) o, se nessuno di questi parametri è prioritario, di considerare la semplicità d'uso, cioè l'efficienza umana. Semplicità d'uso significa che uno degli ostacoli principali nell'impiego dei microprocessori in un nuovo progetto non è la sua complessità tecnica. Si mostrerà nel prossimo capitolo che mettere insieme un sistema a microprocessore è semplice. Generalmente il vero ostacolo risiede nella programmazione dell'intero sistema. Per questa ragione la scelta del microprocessore adatto potrebbe arrestarsi sulla semplicità o facilità (o difficoltà) di eseguirne la programmazione. Un importante criterio di scelta sarebbe allora: ho bisogno di un linguaggio ad alto livello oppure ho bisogno della compatibilità con un linguaggio o con un set di istruzioni che io già conosco? Se le cose stanno a questo modo la scelta si riduce di parecchio. Altrimenti, e in particolare se l'utilizzatore è esperto nell'uso del linguaggio assembler, qualsiasi microprocessore sarebbe fondamentalmente altrettanto difficile o duro da usare e questo criterio non risulterebbe significativo. Un altro criterio importante in termini di semplicità di impiego è: è disponibile un sufficiente supporto? Si mostrerà al Capitolo 9 che il supporto include possibilità di hardware e di software. Supporto significa altresì che un vantaggio significativo è la disponibilità di personale familiarizzato con quel componente, facilmente accessibile all'utilizzatore. Ciò significa disponibilità di ingegneri di supporto o di individui con esperienza appartenenti a gruppi circostanti che possono dare un

aiuto ai suoi sforzi. Questo fatto può significare la differenza tra un successo e un fallimento di uno sviluppo che debba essere terminato in un tempo limitato.

Per poter trarre il massimo vantaggio da questo capitolo il lettore dovrebbe comprendere sia gli aspetti hardware sia quelli software dei microprocessori. Questo implica la comprensione dei capitoli riguardanti l'interconnessione del sistema, il suo interfacciamento, il suo software e il sistema di sviluppo. Si suggerisce perciò di leggere questo capitolo una seconda volta alla fine del libro. A quel punto, dovrebbe contribuire significativamente a chiarire le importanti similarità e differenze tra i prodotti oggi sul mercato e condurre così ad una più facile scelta.

CAPITOLO 5

COME CONNETTERE INSIEME UN SISTEMA

OBIETTIVO

A questo punto sono stati descritti tutti i componenti necessari per assiemare un sistema. In questo capitolo essi verranno interconnessi e verrà così costruito un sistema completo. Per prima verrà assemblata un'unità centrale di elaborazione (central processing unit, CPU) e verrà interfacciata ad essa una memoria. Verranno poi collegate delle interfacce di input-output. A questo punto sarà stato messo assieme un «sistema microcomputer standard». Nell'ambito di questo capitolo i principi che verranno dapprima presentati saranno immediatamente seguiti da degli esempi applicativi reali. Costruiremo un sistema standard a microprocessore e poi esamineremo la reale interconnessione di un certo numero di sistemi reali.

Si mostrerà che non è necessaria alcuna conoscenza dettagliata di hardware al di là di quanto detto per assiemare un microcomputer. «Un bambino può farlo» (e lo fanno realmente). Alla fine di questo capitolo il cablaggio completo di un sistema dovrebbe risultare semplice.

ARCHITETTURA STANDARD DEL SISTEMA

L'architettura del nostro sistema microprocessore «standard» compare in Fig. 5-1. Si mostrerà in un capitolo successivo che i sistemi «non standard» consistono normalmente solo in variazioni minori di questo schema.

Il sistema è caratterizzato per prima cosa dai tre bus standard: il data bus bidirezionale a 8 bit, l'address bus monodirezionale a 16 bit e il control bus che sono creati dal microprocessore. Tutti i componenti del sistema standard sono collegati a questi 3 bus. La CPU, sulla sinistra, include il microprocessore più il clock e il quarzo richiesti. Può richiedere in aggiunta dei bus driver per i tre bus (che non sono mostrati in questa figura). I componenti del sistema standard che verranno interconnessi sono la memoria (ROM o RAM) e le interfacce standard di input e di output: la UART (per la conversione serie-parallelo) e il PIO (per le interfacce parallelo). Per periferiche specifiche saranno richiesti altri componenti specializzati. Inoltre il sistema potrebbe richiedere dei chip di sequenzializzazione come un PIC per gestire le priorità o di un DMA per il trasferimento automatico dei blocchi o di un PIT per una temporizzazione precisa di intervalli di tempo. Infine sarà richiesta un'alimentazione per il funzionamento del sistema completo e questa compare sulla

sinistra della figura.

Tutti gli altri componenti LSI verranno interfacciati sia direttamente ai bus, sia a questo gruppo base di componenti LSI.

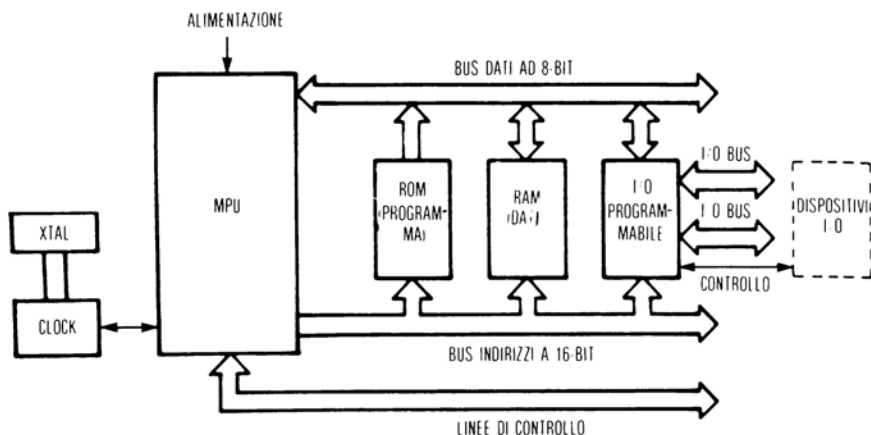



Fig. 5-1: Sistema a microprocessore standard.

Questo sistema standard verrà ora esaminato in dettaglio puntualizzando le opzioni disponibili e i problemi di interconnessione.

MONTAGGIO DI UNA CPU

Una *central-processing-unit* consiste di una ALU più una control unit (CU). Fino alla fine del 1976, una CPU richiedeva almeno un'unità a microprocessore (una MPU) più un clock esterno, più un quarzo, più qualsiasi altro «circuito di supporto» applicabile (come ad esempio i bus driver). Dalla fine del 1976 il circuito di clock in particolare è ormai integrato entro la MPU stessa. Tuttavia l'architettura risultante e, in particolare, i tre bus standard, resta invariata per il nostro sistema standard.

Come esempio di una CPU fondamentale, in Fig. 5-2 appare la interconnessione dettagliata di una CPU con l'8080. Per realizzare una CPU l'8080 richiede almeno tre chip LSI. Essi sono: il microprocessore 8080, il clock 8224 (più il suo quarzo) e il controller di sistema 8228. Ricordiamo che l'8080 fu uno dei primi microprocessori potenti a 8 bit ad essere introdotto e che esso richiede tre alimentazioni (+5, −5V, +12V). Inoltre il suo clock richiede diversi pin. Di conseguenza non rimangono pin in numero sufficiente per permettere di inviare in uscita mediante gate il control

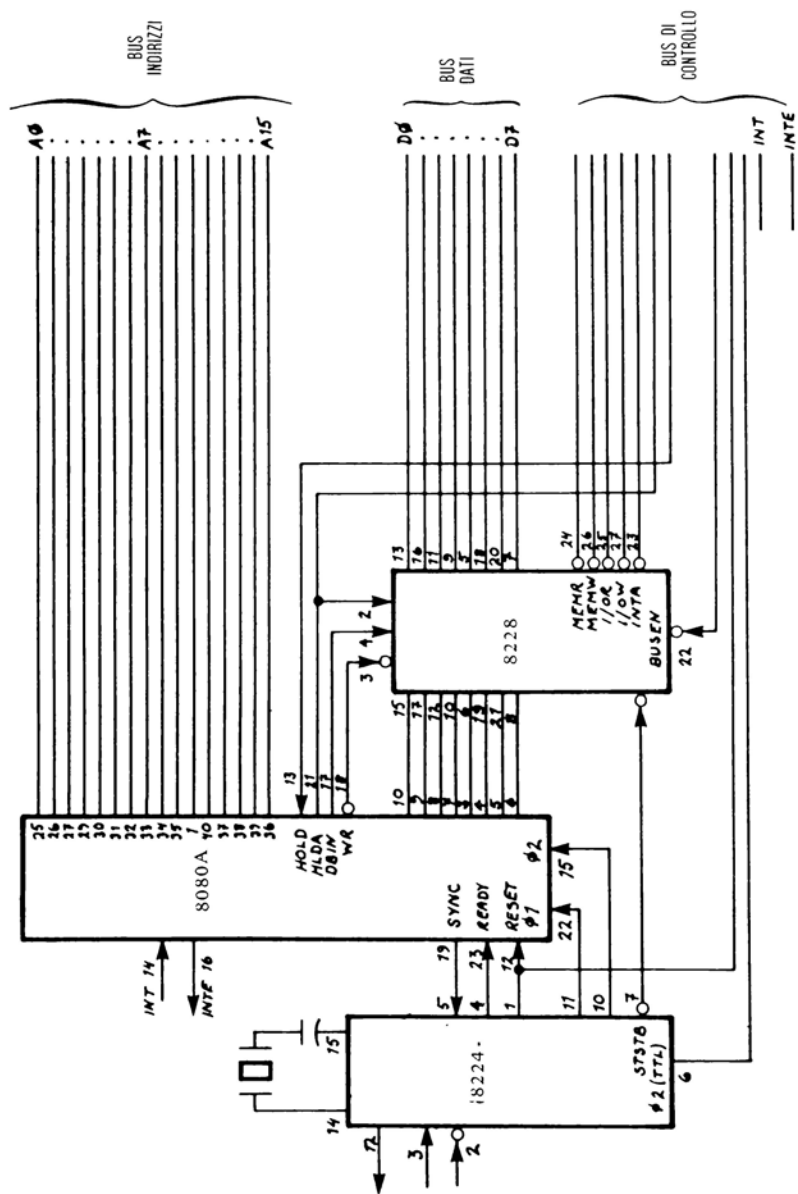



Fig. 5-2: I 3 chips richiesti dalla CPU dell'8080.

bus. La limitazione a 40 del numero dei pin rende necessario multiplexare il control bus sul data bus. Durante lo stato T1 di ciascun ciclo macchina il data bus trasmette l'informazione di stato sui propri pin. Ciò implica la necessità di demultiplexare il data bus esternamente e di porre dei latch su questi segnali di controllo. Questi segnali appaiono in Tabella 5-3. La disponibilità di otto pin sul data bus permette la possibilità di propagare esternamente otto segnali di controllo. In pratica vengono usati solo cinque di questi segnali. L'8228 fornisce cinque segnali di controllo «puliti» per il control bus esterno: MEMR; MEMW (read e write nella memoria o via I/O); I/OR e I/OW (read e write per i dispositivi di I/O); INTA (riconoscimento dell'interrupt). Una caratteristica importante dell'8228 è quella di incorporare un driver per il data bus. L'8080 ha un driver «libero» sul suo data bus per merito dell'8228 (il 6800 necessita di un bus driver nello stesso tempo). Ne consegue che il totale di chip per un 8080 o per un 6800 è identico. Il costo dell'8228 è naturalmente più alto del costo di un semplice bus driver. Ma la necessità dell'8228 non viene sentita come uno svantaggio significativo. Tuttavia si può prevedere che i dispositivi che succederanno all'8080 incorporeranno l'8228 assieme ad un driver del clock su un unico chip. Ciò è già stato fatto nel caso dello Z80 dalla Zilog e per l'8085 dall'Intel.

DB	SEGNALE	SIGNIFICATO
0	INTA	RICONOSCIMENTO INTERRUPT
1	$\overline{WO}$	SCRITTURA O USCITA LETTURA O INGRESSO
2	STACK	SP È SUL BUS INDIRIZZI
3	HALT	RICONOSCIMENTO HALT
4	OUT	INDIRIZZO DEL DISPOSITIVO D'USCITA SUL BUS INDIRIZZI
5	M 1	MPU NEL MODO FETCH
6	INP	INDIRIZZO DEL DISPOSITIVO D'INGRESSO SUL BUS INDIRIZZI
7	MEMR	IL BUS DATI AVRÀ I DATI DI MEMORIA

Fig. 5-3: Sul bus dati (8080) compaiono 8 segnali di stato.

COLLEGAMENTO DELL'ADDRESS BUS

Il principio di base da ricordare è che l'address bus viene usato per *selezionare un registro* all'interno di un componente. Per compiere questa funzione si devono compiere due selezioni:

1. Deve essere selezionato il dispositivo.
2. Deve essere selezionato il registro entro il dispositivo.

Per eseguire la funzione di indirizzamento vengono usate sostanzialmente due tecniche. Inoltre vengono realizzate sui dispositivi stessi diverse tecniche di decodifica interna sia per il «chip-select» che per il «register select». Dal punto di vista del collegamento dell'address bus le due tecniche fondamentali sono la *selezione lineare* e *l'indirizzamento decodificato.*

Selezione lineare

Nella *selezione lineare* una linea dell'address bus viene usata direttamente per *selezionare un componente.* Esaminiamo le conseguenze di questa tecnica. Nella pratica la maggior parte dei sistemi richiederà al massimo una memoria di 4K. La possibilità di selezionare 4K indirizzi entro la memoria implica che 12 bit del campo di indirizzamento dovranno essere riservati per questa funzione ($2^{12}=4K$). Gli altri 4 bit verranno utilizzati come chip select. Un bit potrà essere allocato per selezionare ROM o RAM. In pratica 512 o 1024 parole di RAM sono in genere sufficienti e virtualmente in tutti i casi le dimensioni della RAM sono più piccole delle dimensioni della ROM: i 12 bit allocati per la selezione di parola forniranno altresì l'indirizzamento necessario entro la RAM, quando questa viene selezionata. Infine tutto questo lascia un massimo di tre bit disponibili. Escludendo uno dei codici, come ad

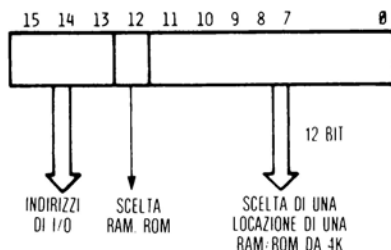



Fig. 5-4: La scelta lineare dedica una linea per ogni dispositivo.

esempio lo 000, per la selezione di memoria, si possono usare 7 combinazioni di questi 3 bit per indirizzare i dispositivi di I/O. Qualsiasi dei 12 bit di «indirizzo» possono venire riutilizzati per selezionare registri entro i dispositivi.

Nonostante le apparenti limitazioni di questa tecnica riguardo al numero di dispositivi indirizzabili, è quella più frequentemente usata nei piccoli sistemi. Nei piccoli sistemi che richiedono solamente una modesta quantità di memoria e un piccolo numero di chip per l'I/O, *non sono necessari dei decodificatori di indirizzo.* I bit 15, 14, 13 ed altri del campo di indirizzo possono essere *connessi direttamente al pin di chip select* del componente specifico. Ciò minimizza il numero di chip e, quindi, il costo.

Se il sistema dovesse successivamente crescere assumendo una dimensione mag-

giore, sarà allora necessario ricablare e riprogrammare e nel contempo si dovranno aggiungere dei decodificatori di indirizzo. Per questa ragione nella maggior parte dei sistemi di medie dimensioni si preferisce includere fin dall'inizio una quantità modesta di decodifica direttamente sulla scheda, così che si possano aggiungere agevolmente in un tempo successivo della memoria aggiuntiva o dei dispositivi. Di converso lo svantaggio di questa tecnica è quello di richiedere inizialmente un numero di componenti più grande di quanto risulta strettamente necessario.

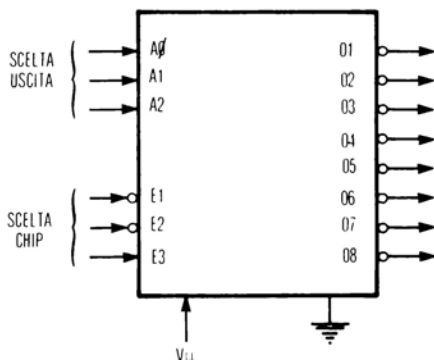



Fig. 5-5: L'8205 è un decodificatore 3 ad 8.

A questo punto le alternative a questa tecnica dovrebbero apparire evidenti. Presenteremo esempi di selezione lineare usata per indirizzare il chip nel paragrafo sull'Interconnessione del Sistema. La presenza di un «1» su una linea di indirizzo verrà interpretata come un chip select da un componente.

Un altro svantaggio della tecnica di selezione lineare è la frammentazione dello spazio di indirizzamento. Ogni volta che viene usata una linea di indirizzo in questo modo lo spazio di indirizzamento viene diviso per due. Ciò conduce a blocchi di indirizzi discreti entro lo spazio di indirizzamento. Lo spreco di spazio potenziale di indirizzamento in memoria non dovrebbe sollevare problemi poiché si è assunto che il sistema non avrebbe richiesto in ogni caso le 2^{16} locazioni. Ciò che può sollevare problemi è il fatto che questi blocchi siano discreti e che la programmazione possa essere resa più difficile a causa dell'attenzione che è necessario rivolgere all'indirizzamento di questi vari blocchi.

La ragione può essere illustrata dall'esempio seguente:

Nella «vera selezione lineare» noi allochiamo un bit per la selezione della RAM ed un bit per la selezione della ROM, ad esempio i bit 13 e 12 rispettivamente. Ne risulta che i possibili indirizzamenti per la ROM andranno da 2048 a 4095 e quelli per la RAM da 4096 a 6143.

Se il bit 14 seleziona un altro dispositivo, l'indirizzo successivo ripartirà da 8K. I blocchi da 0 a 2047 e da 6144 a 8191 saranno inutilizzati. Essi costituiranno dei «buchi» nello spazio di memoria che corrisponderanno al bit 12 = 0 e al bit 13 = 0, rispettivamente.

Indirizzamento decodificato

Con la tecnica dell'*indirizzamento decodificato* un certo numero di linee verrà collegato ad un decodificatore che selezionerà poi linearmente uno qualsiasi dei componenti necessari. Ad esempio in Fig. 5-5 è rappresentato un 8205. Esso accetterà tre input e selezionerà una delle otto possibili uscite in funzione della configurazione di ingresso. Nei sistemi con l'8080 gli 8205 sono ampiamente usati. Nell'esempio di Fig. 5-6 questo dispositivo selezionerà una ROM 8708 oppure un paio di RAM 8111 (queste RAM hanno ampiezza per quattro bit e ci vogliono due chip per fornire una parola di 8 bit).

Se è necessario utilizzare il massimo spazio di indirizzamento, cioé la capacità di indirizzare fino a 64K dispositivi, i decodificatori (decoder) richiesti aumenteranno fino a costituire un insieme molto complesso. Questo di fatto è l'approccio scelto nel caso dei calcolatori standard come anche in molti minicomputer. La complessità derivante dall'uso dei decoder e il relativo numero di chip necessario sono un elemento disincentivante rispetto ad un indirizzamento dei microprocessori completamente decodificato. Sono state introdotte diverse tecniche in alternativa. La tecnica più usuale consiste nel realizzare una decodifica solamente parziale, come ad esempio la decodifica di tre bit dell'8205 che è stata presentata, combinandola con l'indirizzamento lineare per altri dispositivi.

È possibile ancora un altro approccio. Esistono oggi diversi componenti muniti di *chip-select multipli* (o di chip-enable). I vantaggi di un chip select *singolo* consistono naturalmente nel minimizzare il numero di pin sul componente. Il vantaggio del chip-select *multiplo*, come ad esempio tre chip-select, consiste nella possibilità di fornire una decodifica automatica di indirizzo *entro il componente stesso*. Se sono disponibili tre pin, allora tre linee dell'address bus verranno connesse direttamente a questi pin. Essi verranno abilitati ad esempio con la combinazione «110». Ciò equivale alla decodifica della combinazione «110» entro il chip. Purché tutti i chip da inserire nello stesso sistema decodifichino una combinazione diversa (questo si può ottenere anche scambiando le linee a rotazione per una data combinazione di pin), potrà scomparire la necessità di una decodifica esterna. Lo svantaggio di questa soluzione sta nell'aumentato numero di pin che conduce ad un certo aumento dei costi dei componenti. Questo approccio è stato usato ampiamente nel sistema 6800: ciascun dispositivo ha 3 o più «CS» ed un sistema di medie dimensioni può essere costruito senza alcun decoder.

Per i sistemi piccoli e medi, questa soluzione sarà particolarmente vantaggiosa. Oggi la maggior parte dei chip di interfaccia con le periferiche nonché le memorie è munita di chip - select multiplo.

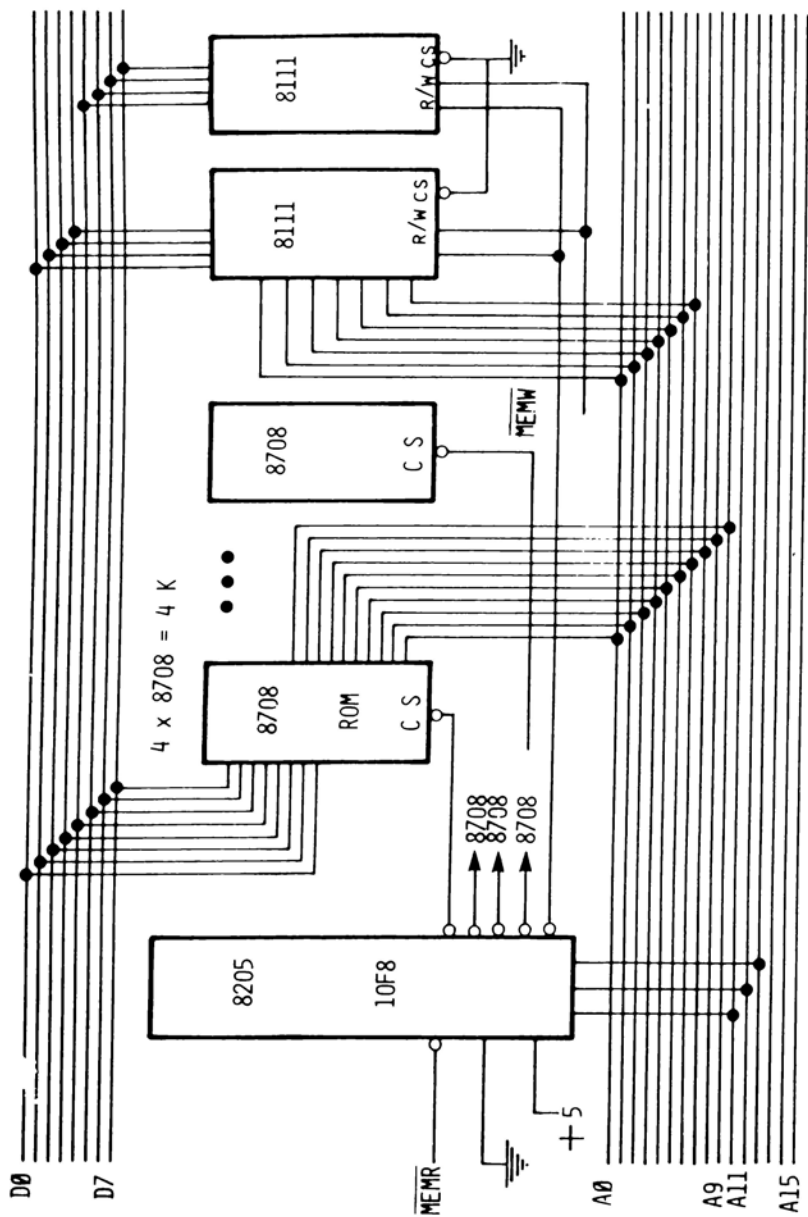



Fig. 5-6: Interfacciamento della memoria ad un 8080 mediante decodificatore.

COLLEGAMENTO DELLA MEMORIA

Nel caso delle memorie è necessario fare un discorso a parte. Poichè le memorie possono espandersi fino ad una grande ampiezza e coinvolgono un grande numero di chip è necessario un criterio di economia. Si potrà economizzare utilizzando il numero di pin minore possibile. Di conseguenza sulle memorie di *impiego generale* (general-purpose) sono realizzati dei chip-select con un pin singolo. Quando si seleziona la memoria è possibile operare diverse altre scelte. Il tipo più economico di memoria fornisce solamente la possibilità di avere un bit di dati in ingresso ed uno in uscita. Ciò chiaramente minimizza il numero dei pin. Per costruire un sistema ad 8 bit si devono connettere *in parallelo* otto chip di memoria di questo tipo. Essi naturalmente verranno selezionati tutti dal medesimo indirizzo. Ciascuno dei loro input - output per il dato verrà connesso ad una delle linee del data bus.

Similmente se viene impiegata una memoria da 4 bit, come ad esempio la 8111, si devono collegare in parallelo i chip due a due (vedere Fig. 5-6).

Se si impiega una memoria da 8 bit, il numero di pin sul package sarà massimo con la conseguenza del più alto costo per bit, ma l'interconnessione risulta più semplice e il numero di chip sarà minore per un piccolo sistema.

DOMANDA: *Perché tutte le memorie standard mettono a disposizione un pin separato di «data input» ed un pin separato di «data output», considerando che tutti i microprocessori hanno un data bus bidirezionale?*

RISPOSTA: *La risposta dovrebbe risultare semplice. Le memorie sono componenti standard che non sono stati previsti solamente per applicazioni su microprocessori. Le memorie sono state concepite per applicazioni generali sui calcolatori. I calcolatori più grandi non sono limitati ad un data bus bidirezionale e normalmente vengono forniti di bus separati di data-in e di data-out. Le memorie devono perciò mettere a disposizione delle connessioni separate di input e di output. Nel caso dei microprocessori i 2 pin sono semplicemente collegati assieme (ad esempio usando 2 transceiver quadrupli).*

Altri pin sul chip di memoria includono: i segnali di read e write, un possibile segnale di sincronizzazione e naturalmente l'alimentazione. La densità corrente dei chip di memoria standard è di 4K bit per chip. Da qualche mese sono disponibili anche 16K bit per chip ma non possono ancora essere considerate altrettanto affidabili fino a quando non sarà passato almeno un anno. Queste densità si riferiscono alle RAM dinamiche standard. Le ROM possono naturalmente raggiungere delle densità più elevate.

ALLACCIAMENTO ALL'INPUT - OUTPUT

La «selezione» dei chip di interfaccia di input-output, ai fini dell'indirizzamento, viene realizzata nel medesimo modo della memoria. Il metodo è stato descritto nel paragrafo precedente. La selezione dei registri entro i chip è ottenuta semplicemen-

te utilizzando l'address bus esattamente nel medesimo modo usato per l'indirizzamento di una locazione di memoria. Se un chip è fornito di otto registri interni, tre pin saranno previsti normalmente per la selezione interna dei registri: questi tre segnali verranno decodificati internamente per selezionare uno degli otto registri. Essi sono connessi all'address bus.

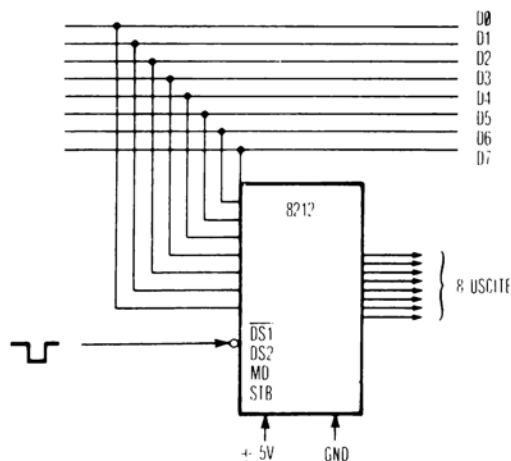



Fig. 5-7: Il latch 8212 è una porta I/O di base.

Mettere a disposizione una possibilità di input o di output richiede, come minimo, un latch. Il collegamento più semplice possibile è illustrato in Fig. 5-7. In pratica è più economico prevedere dei mezzi bidirezionali usando un PIO. Questo

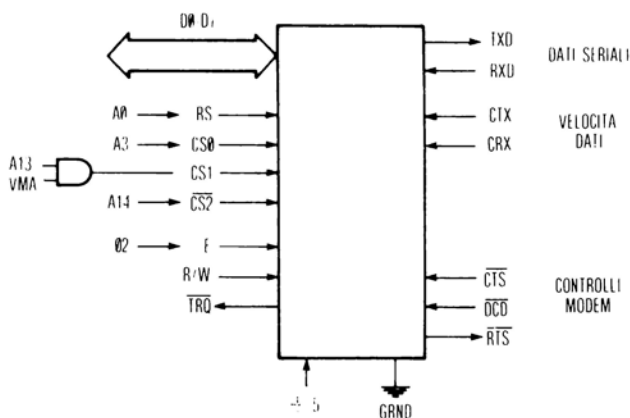



Fig. 5-8: Connessione dell'ACIA (Motorola).

componente è stato descritto nel Capitolo 3. Nel progetto del nostro sistema standard collegheremo esclusivamente dei PIO o delle UART ai bus standard del microprocessore. Il collegamento di questi dispositivi implica una connessione di 8 bit al data bus ed il richiesto collegamento all'address bus per il «chip select» ed il «register select» come spiegato precedentemente. Inoltre esso richiede un collegamento al control bus. La connessione particolare è specifica rispetto al microprocessore usato. Non esiste una regola generale. Tutti i dispositivi saranno connessi ai pin di read e/o write al fine di specificare le operazioni di read o di write entro i registri. Es-

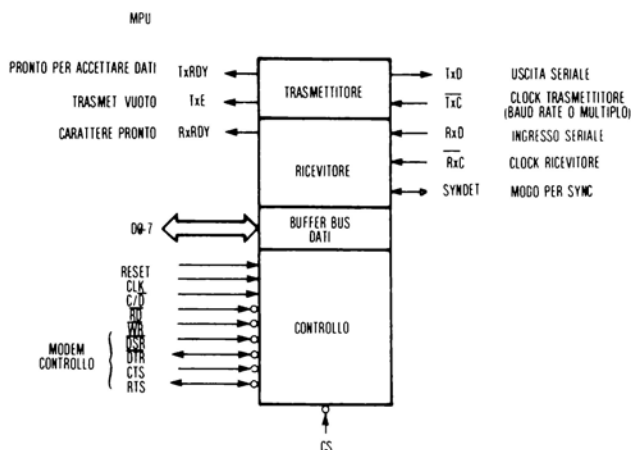



Fig. 5-9: Connessione di un 8251 (Intel).

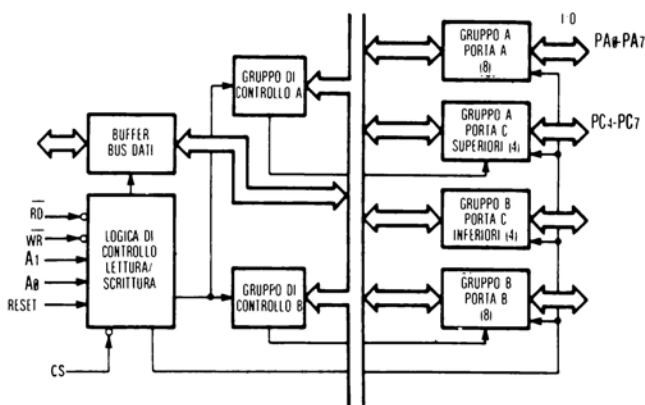



Fig. 5-10: Connessione ,i un 8255 PPI (Intel).

si saranno collegati ad un RESET al fine di azzerare i registri; al clock e alle linee di interrupt se vengono usati gli interrupt. A fini di sincronizzazione possono essere richiesti dalla MPU specifica degli altri collegamenti.

Per illustrare la semplicità dei collegamenti sono stati riportati un'ACIA (la UART della Motorola) ed un 8251 (Intel), che compaiono rispettivamente nelle figure 5-8 e 5-9. Si può vedere che essi sono collegati come ci si aspettava al data bus, all'address bus, per il chip select ed il register select ed al control bus, come indicato.

Similmente l'interconnessione di un 8255 (PPI) compare nell'illustrazione successiva ed ogni spiegazione è conseguente.

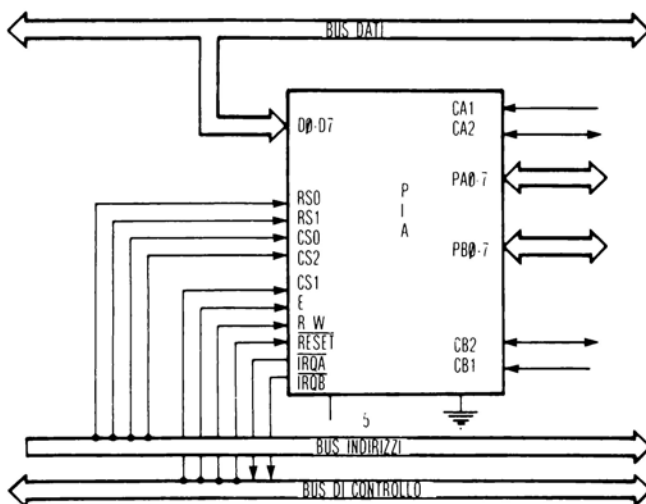



Fig. 5-11: Connessione di un PIA 6820 (Motorola).

INTERCONNESSIONE DEL SISTEMA

L'interconnessione effettiva del nostro sistema standard a microprocessore appare in Fig. 5-12. Essa si riferisce al microprocessore standard monolitico a 8 bit che è stato già descritto.

In microprocessori specifici si potranno riscontrare delle variazioni di minor importanza; ad esempio alcuni microprocessori mettono a disposizione solamente 15 bit invece di 16 nell'address bus. Tuttavia le funzioni più importanti si possono tro-

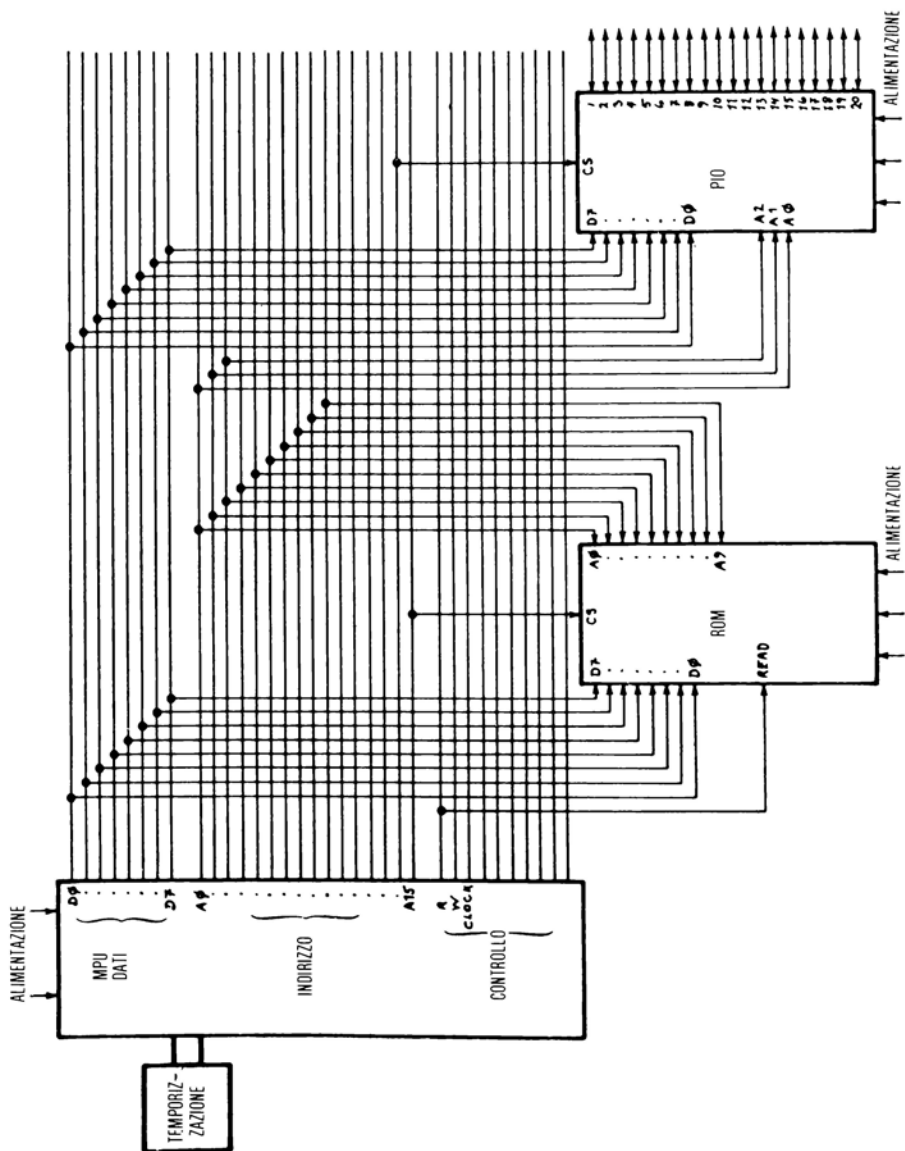



Fig. 5-12: Interconnessione del sistema standard.

vare nel control bus. Le funzioni delle linee relative sono fondamentalmente specifiche del microprocessore considerato. Per questa ragione solo alcune delle linee di controllo sono state etichettate per maggior chiarezza. L'interconnessione verrà ora esaminata con maggior dettaglio.

Dall'alto al basso compaiono i tre bus: il data-bus, l'address-bus ed il control-bus. Le linee sono state mostrate in modo esplicito. Al sistema sono collegati due dispositivi essenziali: la ROM per la memoria e il PIO per l'input-output. Altri dispositivi che normalmente sarebbero collegati sono una RAM ed eventualmente una UART. E ne verrebbero collegate nel medesimo modo.

Esaminiamo i collegamenti. Viene impiegata la relazione lineare poichè questo è un piccolo sistema. Il pin di chip select (CS) può essere individuato in cima ai chip della ROM e del PIO. La ROM è selezionata dalla linea A15 dell'address bus. Il PIO è selezionato dalla A14 dell'address bus. Si assume che questa sia una ROM da 1Kx8. Sul lato sinistro della ROM compare il collegamento al data bus di 8 bit. Sul lato destro della ROM compare il collegamento all'address bus. L'indirizzamento di 1K richiede 10 bit e vengono stabiliti per questo i collegamenti alle linee da A0 fino al A9 dell'address bus. Inoltre la ROM richiederà un segnale di sincronizzazione che potrebbe essere la linea di READ che è mostrata in figura. Naturalmente, essa non richiede una connessione alla linea WRITE, essendo questa una memoria di sola lettura. Infine l'alimentazione (power) compare in fondo alla figura.

Se questa ROM avesse avuto l'ampiezza di soli 4 bit, avremmo collegato semplicemente due di questi componenti in parallelo.

Il PIO è selezionato da A14. Si assume che esso sia fornito al massimo di otto registri interni. Questi registri sono selezionati semplicemente da A0, A1, A2 dell'address bus e questi collegamenti compaiono sul lato sinistro del PIO. A questo punto sorge un conflitto apparente: A0, A1, A2 sono collegate sia al PIO che alla ROM. Entrambi i chip verranno abilitati contemporaneamente? Naturalmente A0, A1 e A2 verranno abilitati sia per il PIO che per la ROM. La differenza sta nel fatto che la ROM verrà selezionata solo quando A15 è pari a 1 e il PIO verrà selezionata solo quando A14 è pari a 1. Queste due eventualità sono mutualmente esclusive con i collegamenti indicati o meglio: è responsabilità del programmatore assicurarsi che A14 ed A15 non vengano abilitate simultaneamente. Il PIO è collegato alle otto linee del data bus e fornisce un numero di linee di input-output che compaiono sul lato destro. Inoltre esso sarà collegato alle linee di READ e di WRITE del control bus che per chiarezza sono state omesse dalla figura. Normalmente esso sarà collegato anche alla linea di interrupt ed eventualmente a qualche altra linea di controllo speciale del sistema.

Il collegamento di una RAM o di qualsiasi altro dispositivo simile è essenzialmente identico al collegamento della ROM eccettuato per la presenza della linea di WRITE. Una RAM richiederà un collegamento ad una linea di WRITE oltre al

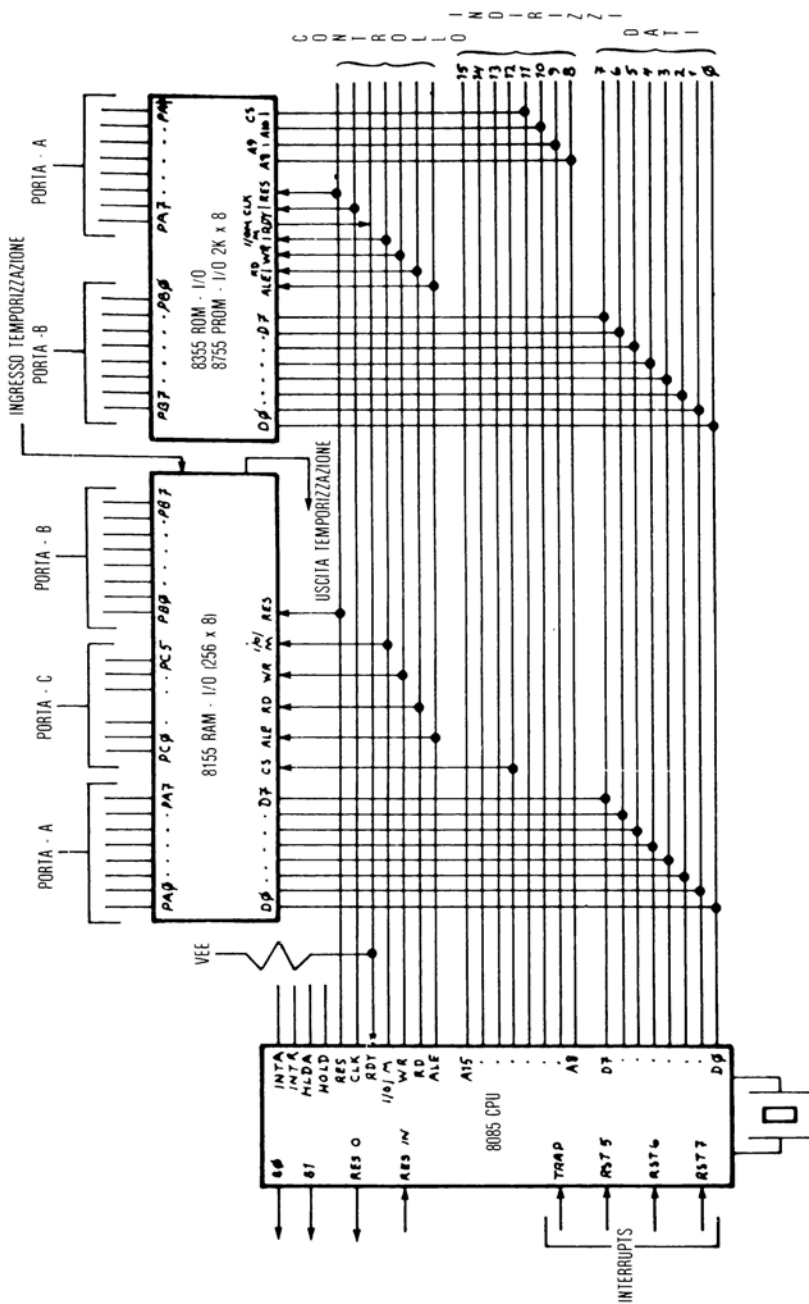



Fig. 5-13: Interconnessione dell'8085.

collegamento della linea di READ. Il resto della decodifica è identico.

L'interconnessione di base di questo sistema dovrebbe ora essere chiara. Potrebbero sorgere dei problemi specifici di interconnessione, dovuti alla natura del control bus a causa di differenti connessioni. Ad esempio l'8080 è fornito di due linee separate di READ e di WRITE. Il 6800 usa una linea unica di READ/WRITE più una linea di validazione indirizzo per la memoria (VMA).

Entrambi richiedono due linee; tuttavia l'uso di queste linee è indifferente. In altre MPU si utilizzano metodi ancora diversi. Ad esempio nella pagina seguente compaiono le interconnessioni di un sistema col 6800. I bus creati dal 6800 compaiono nell'illustrazione successiva.

Un sistema con l'8085

In Fig. 5-13 compare l'interconnessione di un sistema con l'8085, come interessante esercizio e come esempio di un altro sistema reale. L'8085 è il «successore» dell'8080. Esso incorpora l'8080, il clock 8224 e il «system controller» 8228 in un unico chip. È compatibile con il set di istruzioni dell'8080. Sfortunatamente l'8085 si è trovato ancora una volta a corto di pin. Questa volta il control bus non è più multiplexato sul data bus. L'address bus dell'8085 ha solamente otto linee che sono usate come porte di uscita da A8 fino ad A15. Le otto linee della porta bassa dell'indirizzo sono da D0 a D7. Una linea speciale «ALE» denota il fatto che un *indirizzo* è correntemente in trasmissione sul data bus al posto di un *dato* (ALE = address latch enable). Il data bus deve essere demultiplexato esternamente e si deve fornire un latch.

Il valore reale dell'8085 risiede nella disponibilità di componenti speciali, l'8155 (RAM più I/O) e l'8355 (ROM più I/O). Poichè i due componenti speciali forniscono, allo stesso tempo, memoria I/O e demultiplexaggio del data bus, un sistema completo può essere assiemato con appena tre chip. Perciò l'8085 è un sostituto ideale dell'8080 per i piccoli sistemi con l'8080. Esso non sostituisce l'8080 nei grandi sistemi.

L'8155 incorpora in un unico chip una RAM da 256x8 e mette a disposizione l'equivalente di un PIO con tre porte esterne (due porte da 8 bit ogni linea della quale può essere programmata indipendentemente come input o output ed una porta da 6 bit, Porta C, normalmente utilizzata per l'hand-shaking). Inoltre esso include un timer.

L'8355 mette a disposizione 2Kx8 di ROM più l'I/O. L'8755 fornisce una PROM più l'I/O ed entrambi forniscono inoltre due porte da 8 bit di input-output.

Esaminiamo ora l'interconnessione effettiva del sistema in figura. L'8085 è munito di cinque linee di interrupt che compaiono sulla sua sinistra. Due pin sono allocati per il cristallo esterno e due pin per l'alimentazione. Sul lato destro dell'8085 si vedono le linee principali del control bus oltre all'address bus e al data bus.

Esaminiamo ora in dettaglio i collegamenti del chip 8155 (RAM più I/O). Proce-

dendo da sinistra a destra si vedono collegate prima le otto linee del data bus e poi il chip select. Qui il chip select è derivato dall'A12 dell'address bus. Questa è proprio semplicemente la tecnica della relazione lineare. Continuando da sinistra a destra si vede il segnale ALE (Address Latch Enable) quindi il segnale READ (RD) e poi il segnale WRITE (WR), poi ancora il segnale I/O/M utilizzato per scegliere tra le I/O e la memoria del componente ed infine il segnale di reset (RES) utilizzato per inizializzare i contenuti dei registri interni a 0. In cima al componente compaiono i tre bus usati per comunicare con i dispositivi esterni di I/O. Infine sul lato destro si vedono le due linee del timer interno usato per misurare i tempi trascorsi in ingresso e uscita.

DOMANDA: *Perchè questo dispositivo non è collegato all'address bus?*

La risposta è che questo dispositivo risulta di fatto collegato all'address bus anche se ciò non sembra ovvio dalla figura. Osservate attentamente. Il dispositivo è collegato da D0 a D7. Questi sono i bit da A0 ad A7 dell'address bus. Come fa il dispositivo a sapere quando i bit presentati da D0 a D7 rappresentano un indirizzo? Lo viene a sapere tramite l'ALE. Ogni volta che ALE è vero (true) gli otto bit sul data bus vengono interpretati come indirizzi ed acquisiti (con un latch) internamente. Ogni volta che ALE non è vero («true» e «not true» sono due stati convenzionali opposti) questi bit sono interpretati come data bus. Poichè questo componente include solamente 256 parole di RAM, sono sufficienti otto bit per selezionare una qualsiasi locazione di RAM entro il dispositivo.

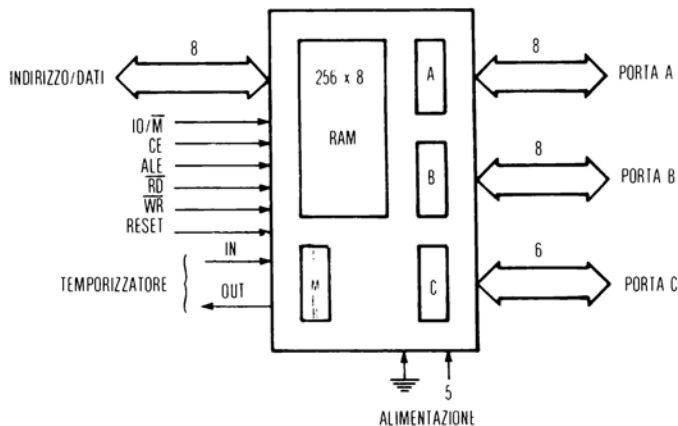



Fig. 5-14: RAM 8155 + I/O.

Un problema ulteriore che non è stato chiarito è come sia possibile selezionare o i registri di I/O oppure la memoria entro il dispositivo. Ciò viene realizzato con il segnale I/O/M. Con I/O si vorrà specificare un accesso ad una porta di I/O e con M si vorrà specificare un accesso alla porzione di memoria del dispositivo. I segnali

di READ o di WRITE specificheranno l'operazione richiesta nel registro o nella memoria.

In modo simile il collegamento alla 8355 (ROM più I/O) dovrebbe essere chiara.

Fig. 5-15: ROM 8755 + I/O.

Questo componente fornisce 2K indirizzi oltre alle porte di I/O. 2K richiedono 11 bit. Infatti esso è collegato ad A8, A9 e A10 dell'address bus. Gli otto bit inferiori sono forniti dal data bus. Il chip select per il componente (CS) viene qui fornito dalla linea A11 dell'address bus. È necessario esaminare anche la ragione per l'utilizzo della linea A11: si utilizza A11 specificamente al fine di fornire un indirizzamento *continuo.* Infine gli altri segnali di controllo che sono collegati al componente sono: ALE, RD, WR, I/O/M, RDY (Ready, un segnale di sincronizzazione), CLK (clock) e RES.

Le linee A13, A14, ed A15 dell'address bus non sono state usate su questo sistema e potrebbero venir usate per collegare dei chip addizionali.

DOMANDA: *Perchè il segnale WR (Write) è collegato ad un 8355 ROM (una Read Only Memory)?*

La risposta dovrebbe essere semplice.

Connessione di altri sistemi

Le interconnessioni di altri sistemi a microprocessore standard non dovrebbero riservare altri misteri. Il cablaggio di altri sistemi che siano differenti in modo significativo include due categorie: i microcomputer a 2 chip ed i bit-slice. L'interconnessione dei bit-slice è stata descritta nel capitolo precedente. Non è compresa negli scopi di questo libro. Riassumendo brevemente l'assemblaggio dell'ALU è semplice è può essere realizzato ponendo in parallelo gli elementi di 4 bit. L'assemblaggio della sezione di controllo completa del programma, handling dell'interrupt, conta-

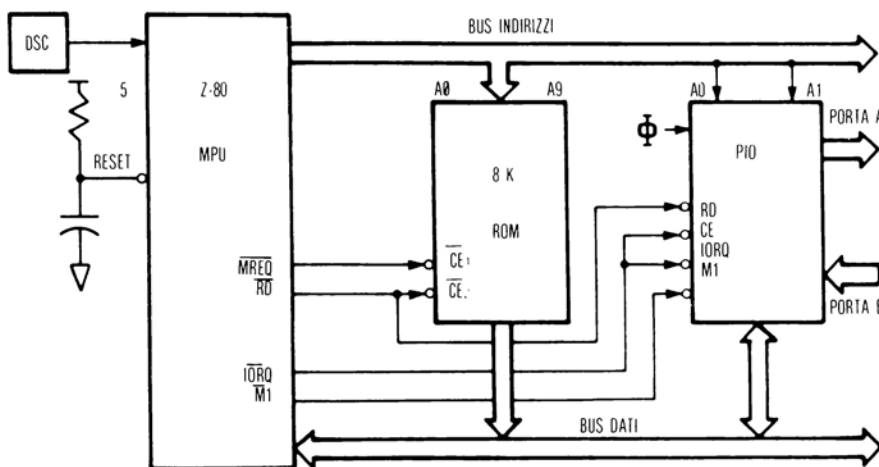



Fig. 5-16: Sistema Zilog-80.

tore di loop, supervisione di condizione e di branching, è complessa è richiede un numero sostanziale di componenti.

Il cablaggio di un microprocessore con 2 chip è essenzialmente una sotto-classe del sistema a 3 chip che è stata presentata nel caso dell'8085. I bus sono semplicemente connessi direttamente assieme per i due chip.

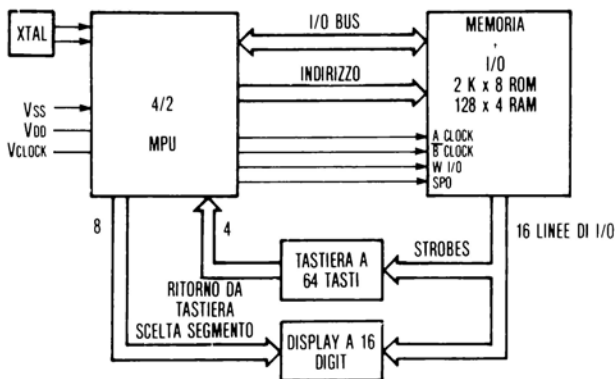



Fig. 5-17: Sistema PPS4/2 della Rockwell.

Esiste un'ultima possibilità di espandere le capacità dei microcomputer su un unico chip. Come esempio in Fig. 5-18 compare l'interconnessione di una memoria addizionale più l'I/O di un 8048. Questa contempla poco più che una interconnessione dei bus.

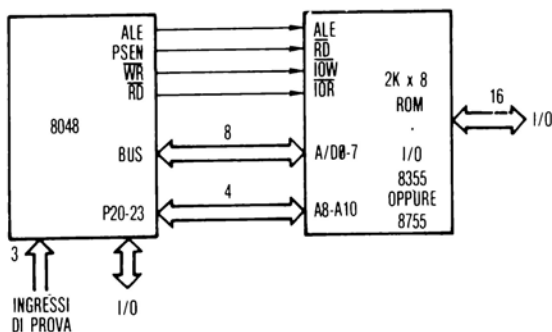



Fig. 5-18: 8048: aggiunta di ROM + I/O.

L'unica ulteriore complessità apparente può risiedere nella connessione di altri componenti «meno standard». Questo punto verrà considerato ora.

Connessione di altri dispositivi

In funzione della filosofia di sequenzializzazione o «scheduling» dell'interrupt che è stata realizzata nel sistema (polling, interrupt o DMA) sorgerà o meno la necessità di collegare chip addizionali. Il collegamento di un PIC è stato già descritto al Capitolo 3. Il collegamento del DMA è stato anch'esso già descritto in quel capitolo mentre l'interconnessione completa di un AMD 9517 compare in Fig. 5-19 nel caso di un sistema con l'8080. Non sorge alcuna complicazione sostanziale.

Il problema successivo sorge in relazione al collegamento di componenti addizionali come dei controllori «device controller» o dei dispositivi di I/O a quelli standard che sono stati descritti (Capitolo 7).

CONCLUSIONE

L'interconnessione dei diversi elementi di un sistema microprocessore è semplice. I componenti LSI hanno già integrati al loro interno gli aspetti più complessi del lavoro di interconnessione. Ne risulta che la costruzione del sistema completo non pone alcun problema significativo. Inoltre risulta che non è possibile cambiare l'architettura di sistema dei microprocessori monolitici. Dato un microprocessore ed il suo chip di supporto, non esiste virtualmente alcuna possibilità di variazioni nell'architettura di sistema risultante al di là dei sistemi multi-microprocessori. Tutti i componenti del sistema sono semplicemente collegati al data bus, all'address bus ed al control bus secondo le tecniche che sono state presentate. Per comprendere con precisione i segnali del control bus è normalmente necessario riferirsi alla descrizione dei segnali dei costruttori. In media l'interconnessione è un processo che può essere implementato rapidamente senza errori e senza un'esperienza precedente significativa. Di fatto ciò ha reso possibile un nuovo mercato, chiamato il mercato degli hobbisti. Studenti, dottori, avvocati ed uomini d'affari, persino senza esperienza precedente di elettronica sono in condizioni di assemblare qualsiasi sistema standard a microprocessore e di farlo funzionare. Gli unici due problemi significativi sono:

1. l'effettivo interfacciamento dei dispositivi di I/O. Questo problema è stato considerato per tutti i dispositivi di input-output e le tecniche di interfacciamento verranno presentate nel Capitolo 7;
2. il secondo problema risiede nella programmazione del sistema ed i principi della programmazione verranno presentati al Capitolo 8.

Ora che è stato assiemato un sistema base possono essere considerate le possibili applicazioni dei sistemi a microprocessore. Queste verranno esaminate al Capitolo 6.

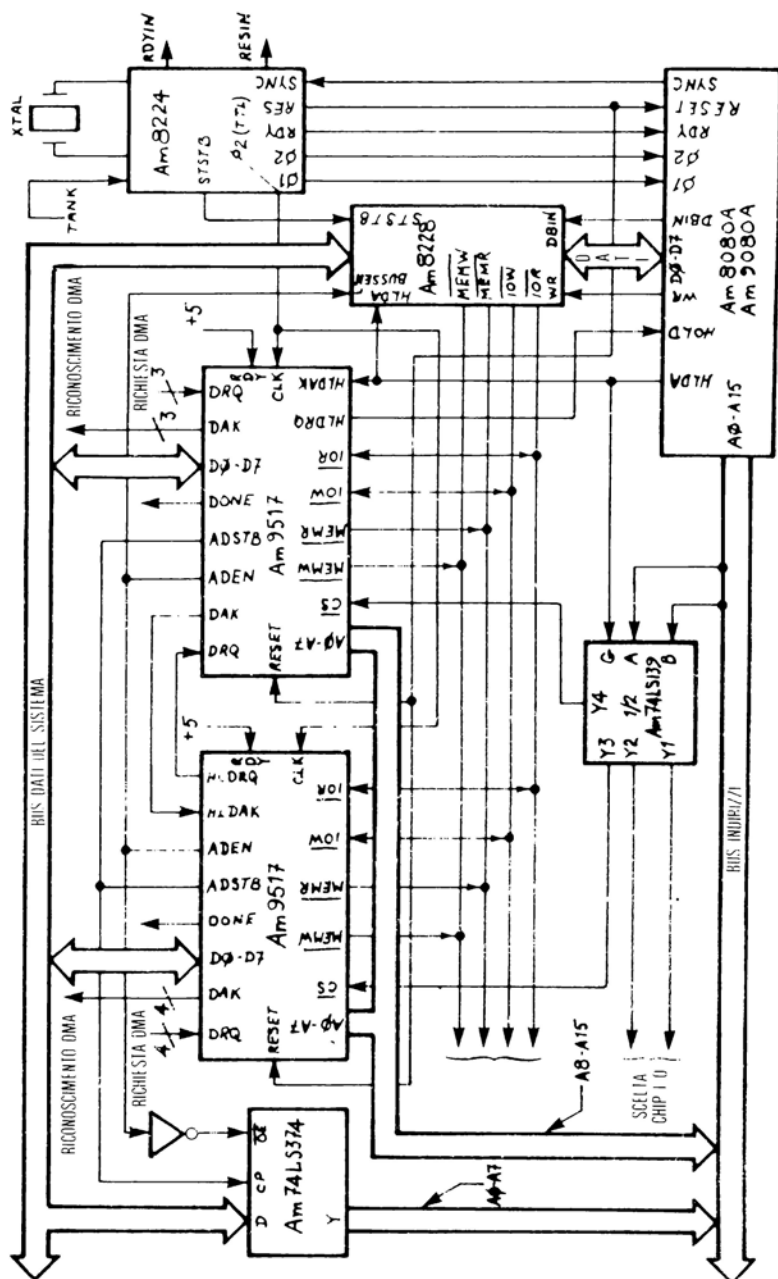



Fig. 5-19: Connessione di un DMAC 9517.

CAPITOLO 6

APPLICAZIONI DEI MICROPROCESSORI

INTRODUZIONE

Questo capitolo presenterà degli esempi di applicazione. Si possono distinguere essenzialmente quattro aree di applicazione per i microprocessori. Inizialmente queste verranno esaminate mettendo in evidenza come esse implichino caratteristiche tecniche diverse. Saranno poi presentati degli esempi specifici. Si mostrerà che l'aggiunta di più funzioni ad un sistema a microprocessore conduce alla semplificazione dei moduli aggiuntivi di hardware e software. L'architettura del sistema rimane essenzialmente invariata. I chips devono essere aggiunti sui bus e possono essere semplicemente interconnessi. Applicazioni progressivamente più complesse saranno realizzate connettendo una quantità crescente di moduli in un progetto standard a microprocessori. La conclusione essenziale che deriverà da questo capitolo è che la realizzazione di un'applicazione a microprocessori è essenzialmente semplice finchè l'applicazione stessa non impone limiti particolari. Un sistema completo può essere assemblato facilmente con i moduli LSI disponibili. I due problemi significativi che rimangono e normalmente richiedono un notevole sforzo di progetto sono l'interfacciamento e la programmazione e verranno considerati nei due capitoli successivi. Si analizzeranno inizialmente le principali aree di applicazione.

AREE DI APPLICAZIONE

Data la facilità di assemblaggio dei moduli di un sistema a microprocessori, l'elenco delle applicazioni possibili è illimitato. Il fattore limitante non è tecnico. Le due principali limitazioni allo sviluppo di una nuova applicazione sono le seguenti: primo, la competenza dell'utente e secondo, la sua immaginazione. Per questo motivo nessuna lista di applicazioni sarà mai esauriente. Comunque in questa sede è conveniente distinguere quattro aree di applicazione principali in corrispondenza a specifiche architetture dei microprocessori. Esse sono:

1. Sistemi di Calcolo
2. Sistemi Industriali
3. Applicazioni Consumer
4. Sistemi per Scopi Speciali (Special - Purpose)

Ciascuna di queste aree di applicazione verrà ora analizzata in maggiore dettaglio e verrà condotta un'analisi delle caratteristiche tecniche.

SISTEMI DI CALCOLO

I microprocessori sono stati inizialmente utilizzati nel campo delle applicazioni di calcolo. La ragione è semplice: l'impiego di microprocessori, specialmente quando essi sono stati introdotti, richiede una notevole conoscenza hardware ed una buona esperienza software. La combinazione di queste conoscenze era più disponibile nei calcolatori commerciali. Non appena i microprocessori furono introdotti i costruttori intravidero l'importanza del loro impiego come elaboratori a costo molto basso per semplici applicazioni di controllo quali il controllo di un dispositivo. Infatti il primo microprocessore ad 8 bit (l'Intel 8008) era nato dall'inconveniente per il costruttore di periferiche di avere un chip microprocessore realizzato per il controllo diretto di un CRT (contratto tra Datapoint, Texas Instruments ed Intel).

I microprocessori sono ora utilizzati per il controllo virtualmente di tutte le periferiche del calcolatore che non richiedono velocità tipiche dei dispositivi bipolari. Inoltre, la recente disponibilità di controllori di dispositivo specializzati, come i controllori CRT e quelli di floppy-disk (FDC) rendono ora possibile l'impiego di micropressori monolitici «lenti» anche in unione con dispositivi veloci come i CRT ed i dischi. Questo è dovuto al fatto che la manipolazione veloce richiesta è eseguita da speciali controllori di dispositivo e da DMAC. Esistono naturalmente ancora alcune eccezioni in cui i microprocessori non sono abbastanza veloci. In particolare i dischi veloci richiedono ancora velocità tipiche di dispositivi bipolari. Tuttavia queste sono eccezioni di rilievo. I microprocessori sono utilizzati per il controllo dei lettori e dei perforatori del nastro di carta, delle stampanti, delle tastiere, dei relé, dei convertitori analogico - digitale, ecc.

Forse è più importante notare che la realizzazione filosofica dei sistemi computer è cambiata. In tutti i sistemi di medio e basso costo (cioè l'equivalente dei primi minicomputers) la stessa capacità di elaborazione è ora disponibile ad un costo di pochi dollari. Essa è divenuta una delle risorse più a buon mercato del sistema. La vecchia filosofia di progetto di calcolatore (e quella ancora impiegata nei sistemi molto grandi dove la CPU rappresenta il costo maggiore) era di dividere le risorse più dispendiose del sistema, cioè la CPU. Ora la CPU è divenuta una delle risorse più a buon mercato del sistema. Sarebbe un errore di progetto di base dividere il tempo dell'elaboratore, cioè multiplexarlo. Questo implica la scomparsa di sistemi operativi ed esecutivi complessi per i piccoli calcolatori. Un traguardo essenziale di questi complessi «esecutivi» era di dividere il tempo del processore in un certo numero di compiti contemporanei in esecuzione parallela. Questo non avviene più. La soluzione più economica, semplice ed efficiente è ora di dedicare un processore al processo. Premesso che il microprocessore non richiede risorse speciali o una grande quantità di memoria, può anche essere possibile realizzare completamente il sistema su un microcomputer su chip singolo al costo da 1 a 10 $. La complessità ed il costo dello sviluppo software per i processori a divisione di tempo è molto maggiore di qualunque altra cosa. Naturalmente esistono alcune eccezioni a questa regola in casi speciali. I casi «speciali» sono essenzialmente caratterizzati dal fatto

che il costo del software non è conteggiato o preso in considerazione (esempi specifici sono lasciati all'esperienza del lettore).

Ora siamo veramente entrati nell'era dei *sistemi distribuiti.* In tali sistemi distribuiti l'intercomunicazione tra un certo numero di processori è ridotta al minimo. Essi non interagiscono in tempo reale ma si scambiano dati, parole o blocchi. Ogni processore è quindi un diretto controllore di processo che controlla completamente un processo. Questi sistemi possono forse non essere qualificati come «sistemi multiprocessori». Essi indubbiamente comprendono multiprocessori; ma in realtà essi non interagiscono tra loro per scopi di controllo in tempo reale.

Si può prevedere che la maggior parte dei sistemi a microprocessori, prima o poi, coinvolgerà più microprocessori. I microprocessori saranno installati nei chips delle periferiche del sistema. Essi possono essere installati nelle unità PIO, UART ed in altri chips del sistema. Inoltre questo renderà la programmazione un compito molto più difficile ed efficace. Ognuno dei chips tradizionali che è usato come semplice dispositivo d'interfaccia diverrà in questo modo interamente programmabile. Le istruzioni saranno inviate a questi dispositivi da un microprocessore, sotto il controllo del programma. Questi processori residenti in dispositivi periferici dovrebbero essere considerati come slaves (schiavi).

MULTIMICROPROCESSORI

Praticamente e concettualmente è semplice interconnettere una rete ad un certo numero di microprocessori. Questa rete verrà allora definita sistema «multimicroprocessore». In tutti i casi dove esiste un processore principale ed un certo numero di microprocessori slave, il sistema verrà definito in questa sede come un «sistema distribuito» piuttosto che, come un sistema multimicroprocessore. Un sistema multimicroprocessore farà riferimento ad un sistema dove un qualunque numero di processori può assumere un ruolo di master. Questo coinvolge un'interconnessione complessa ed un sistema operativo altamente complesso per sincronizzare il funzionamento dell'intero sistema. Si può affermare che la filosofia stessa di un tale sistema sia opposta all'efficienza-costo. In qualunque realizzazione che tenga conto preliminariamente del costo, questa filosofia è semplicemente irragionevole. Il costo di sviluppo di un complesso sistema operativo in grado di sincronizzare un certo numero di processori funzionanti contemporaneamente è impressionante, come è stato dimostrato su un lungo periodo di anni. Inoltre esso implica un alto livello di non affidabilità del sistema, come pure non può essere dimostrato che il sistema funzionerà correttamente in tutte le circostanze. Il costo ed il rischio coinvolti nello sviluppo di un software talmente complesso è totalmente irragionevole se confrontato al costo dell'hardware impiegato. La filosofia corretta è quindi di semplificare e distribuire le risorse hardware dove necessario per un'efficiente realizzazione software. In altre parole la corretta filosofia di realizzazione è di utilizzare processori dedicati dove sono richiesti, in loco, e di sincronizzare il loro funzionamento nel modo più libero possibile per semplificare il funzionamento del sistema globale.

Questo conduce a costo più basso, semplicità e più elevata affidabilità. Esiste un'eccezione a questa esigenza. È indubbiamente normalmente irragionevole considerare i sistemi multimicroprocessori comprendenti microprocessori monolitici per la ragione appena indicata. Comunque è perfettamente ragionevole ed altamente desiderabile considerare realizzazioni multiprocessori impiegando dispositivi bit-slice. I dispositivi bit-slice, sono stati specificamente creati per funzionare facilmente in parallelo. È possibile immaginare e progettare un certo numero di nuove architetture di computer impiegando un certo numero di slices funzionanti contemporaneamente su varie correnti di istruzione. Infatti questa è la corretta filosofia di progetto per migliorare la velocità in qualsiasi dispositivo di elaborazione dati. Lo scopo dei bit-slices esula da questo libro e l'argomento non sarà trattato ulteriormente in questa sede. Esisterebbero una o più eccezioni alla precedente richiesta per qualsiasi sistema che potrebbe essere ripetuto un grande numero di volte in modo da distribuire il costo del software su un grande numero di unità prodotte. In vista della grande complessità di un sistema multimicroprocessore, tale diffusione su larga scala è improbabile. In breve si ritiene che sia «contro natura» considerare i sistemi a multimicroprocessori monolitici (escludendo specificamente i *sistemi distribuiti*).

I sistemi multimicroprocessori sono utilizzati in casi speciali come applicazioni militari ed avioniche, dove il costo non è un ostacolo e dove la microminiaturizzazione è essenziale. Alla fine di questo capitolo saranno presentati dei meccanismi di comunicazione tipici per i multimicroprocessori.

SISTEMI INDUSTRIALI

Le applicazioni industriali di microprocessori corrispondono essenzialmente alla sostituzione di minicalcolatori o di complessa logica cablata con microprocessori di basso costo. Si vedrà che il principale impatto di microprocessori nel mondo industriale non è quello di ridurre il prezzo unitario dell'hardware, almeno in serie di produzione limitate. Il principale impatto è stato quello di fornire un certo numero di nuove funzioni che rendono il controllo del processo più semplice, molto più potente e più «intelligente» pur conservando lo stesso costo. Come beneficio secondario, i microprocessori hanno introdotto il vantaggio del software nel mondo hardware e reso possibile l'impiego di prodotti standardizzati su un periodo di tempo molto più lungo. A sua volta questo si risolve in un costo significativamente più basso, una volta che è stato prodotto un numero sufficiente di sistemi saranno presentati studi di casi specifici.

Le applicazioni industriali di microprocessori sono caratterizzate dalle seguenti caratteristiche tecniche:

La maggior parte delle applicazioni industriali richiedono ingressi ed uscite analogici. Il sistema processore risultante è l'equivalente del controllore analogico con un certo numero di *cicli di controllo*. Un ciclo di controllo è semplicemente la realizzazione di un algoritmo che regolerà un'uscita in funzione di uno o più ingressi.

La maggior parte delle applicazioni industriali sono caratterizzate da costosi sensori e meccanismi di controllo. Il costo dei sensori richiesti per l'ingresso è tale che il costo del sistema microprocessore standard è tipicamente molto piccolo se confrontato al costo di questi dispositivi. In vista del costo globale della capacità di controllo del processo, il basso costo del microprocessore stesso non è un vantaggio significativo. Il vantaggio essenziale è la disponibilità di software dove prima era richiesto hardware. I vantaggi risultanti sono quelli della logica programmata.

La programmazione consente la realizzazione di funzioni di complessità illimitata che in precedenza non potevano essere ottenute con la logica cablata. Le variazioni sono semplici, gli algoritmi possono essere migliorati od anche sostituiti con variazioni hardware minime. Questi vantaggi saranno documentati nel Capitolo 8. I microprocessori, sono utilizzati per controllare processi, flussi, sia discreti che continui, (dal controllo del traffico alla distribuzione d'acqua attraverso condotte), nelle telecomunicazioni e per la regolazione di virtualmente qualunque controllo di processo.

Un tipico controllo di processo può essere la regolazione di un reattore di fermentazione: il microprocessore è equipaggiato con sensori della temperatura, pressione ed altri parametri del sistema, come il PH, la velocità del fluido e le misure di fluidi o gas d'ingresso. Impiegando l'informazione fornita dai sensori, il sistema di controllo osserva il sistema e regola i parametri di controllo che ottimizzeranno la reazione. Esso controllerà, per esempio, la temperatura, la pressione ed i flussi di liquidi e gas, ottimizzando l'andamento del processo. Inoltre esso migliorerà l'affidabilità rivelando o correggendo malfunzionamenti. La pura possibilità di raccolta dati può risultare in un miglioramento della strategia del controllo.

Ogni volta che un microprocessore viene utilizzato per controllo di processo, un vantaggio risultante è la disponibilità del processore per altre funzioni che in precedenza non erano normalmente disponibili. Diviene estremamente semplice aggiungere al sistema una memoria di massa come un registratore a cassette oppure un floppy-disk che consenta di registrare i dati con continuità. Nei periodi di inattività, oppure ad intervalli, regolari, il microprocessore può non solo controllare lo stato del sistema, ma anche registrare tutti i paramentri sulla sua memoria di massa per riferimenti futuri. Esso può anche utilizzare quest'informazione storica per verificare e migliorare l'esecuzione del processo. Questa è chiamata *ottimizzazione dinamica*. Il sistema consulterà i precedenti valori dei parametri di controllo che sono stati trovati dopo un miglioramento del funzionamento del sistema e tenterà di migliorarlo ulteriormente provando nuove alternative.

Le applicazioni industriali sono caratterizzate da tecniche software specifiche non sono usate universalmente.

Retrazione di stato

Ogni sistema di controllo industriale deve essere equipaggiato di una *retroazione di stato* per un funzionamento affidabile. Un microprocessore controlla direttamen-

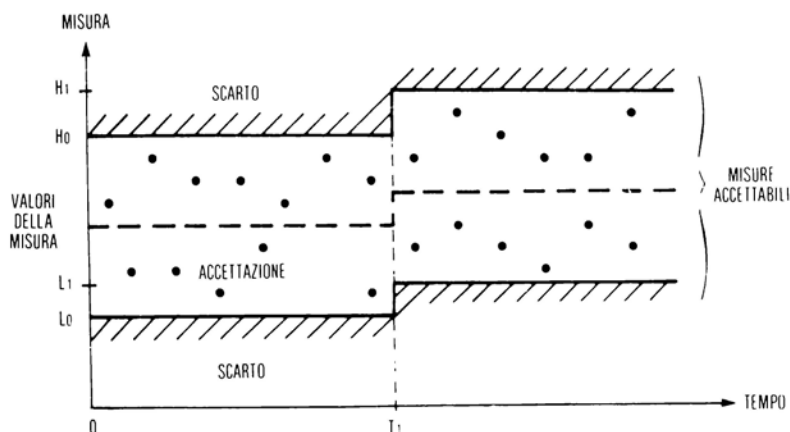
te alcuni dispositivi d'uscita. Esso deve quindi verificare che il funzionamento di questi dispositivi sia corretto. Ogni volta che viene inviato un ordine al dispositivo d'uscita, per esempio «chiudi il relé A», il microprocessore deve verificare che il relé A sia stato chiuso. Ogni dispositivo di controllo deve essere ispezionato in questo modo è deve fornire l'informazione di stato. Lo stato verrà rinviato al microprocessore che lo verificherà. Questo è l'*anello di retroazione di stato.*

Per esempio, un microprocessore darà l'ordine «chiudi il relé A». N millisecondi dopo quest'ordine esso leggerà lo stato di questo relé. Se chiuso correttamente il bit di stato sarà 1 ed il microprocessore determinerà che il relé è stato chiuso correttamente. Se lo stato non devesse essere 1, questo indicherebbe un malfunzionamento. Normalmente il microprocessore darebbe semplicemente l'ordine una seconda e forse una terza volta. Se allora il relé dovesse chiudersi l'informazione di stato direbbe al microprocessore che l'ordine è stato eseguito. Tipicamente il «malfunzionamento» sarà allora considerato come «rumore» ed ignorato e l'esecuzione procederà. Se questo «rumore» si dovesse verificare ripetutamente, il microprocessore suonerà un allarme, richiedendo una manutenzione preventiva. Se il relé non si dovesse chiudere immediatamente, si possono avere diverse alternative. Questo è detto «guasto - leggero». Le tecniche di guasto - leggero coinvolgono una degradazione progressiva del sistema, piuttosto che un guasto completo ogni volta che si guasta uno dei componenti del sistema. In uscita è normalmente difficile rimediare il guasto di un dispositivo di controllo in quanto, essendo dispendiosi, normalmente non sono duplicati. Idealmente il microprocessore dovrebbe attivare un dispositivo di controllo alternativo. Alla peggio, esso eseguirà un algoritmo di ritorno ignorando questo dispositivo di controllo e suonando un allarme esterno. Un controllo simile deve essere eseguito sull'ingresso. La tecnica è diversa. Essa è chiamata «testing di ragionevolezza».

Testing di ragionevolezza

Ogni volta che i valori dei sensori d'ingresso sono letti dal microprocessore, esso dovrebbe determinare se sono corretti. Viene impiegato il testing di ragionevolezza. Normalmente per ogni parametro d'ingresso viene assegnato un intervallo per un istante generico del sistema. Per esempio, un sistema di controllo del traffico urbano ad un incrocio rileverà le automobili sopraggiunte attraverso rilevatori ad anello. Esso utilizzerà l'informazione dei rilevatori ad anello per calcolare la velocità del veicolo. Una velocità di 300 km/h in ambiente cittadino sarà considerata «irragionevole». Questo indica un guasto del meccanismo d'ingresso. Analogamente in un controllore di processo, misurante una temperatura esterna, si possono facilmente rilevare livelli di temperatura irragionevoli.

La procedura usuale è la seguente: ogni volta che si verifica un valore d'ingresso «non ragionevole», esso sarà semplicemente ignorato. Questo viene chiamato «rumore». Questa semplice tecnica esegue infatti un *filtraggio* di indicazioni d'ingresso spurie. Naturalmente, se il guasto dovesse ripetersi, esso indica un malfunziona-

mento del dispositivo d'ingresso. In questo caso verrà generato un diagnostico ed il dispositivo può essere disconnesso dal sistema. È interessante notare che il dispositivo può essere riconnesso automaticamente una volta che il suo ingresso diviene ancora ragionevole. Potrebbe esistere un malfunzionamento *temporaneo*. Il microprocessore conserverà il controllo del dispositivo d'ingresso. Quando il dispositivo d'ingresso fornisce ancora indicazioni ragionevoli per un periodo abbastanza lungo di tempo, esso sarà considerato «ragionevole» nuovamente e sarà automaticamente riconnesso al sistema. La connessione e sconnessione di questo dispositivo al sistema può non essere realizzata fisicamente in hardware. Questo è normalmente eseguito interamente per via *software*. Questa è una tecnica di *peso di confidenza* secondo la quale ogni sensore riceve un rapporto di confidenza o peso. Una misura è ottenuta da diversi sensori d'ingresso «equivalenti» e quindi moltiplicata per questo peso per calcolare il valore d'ingresso finale mediato.

I microprocessori accettano solo L   misura   H e segnalano i valori rigettati

Testing di ragionevolezza

Per esempio, possono essere disponibili due sensori di temperatura. Uno avrebbe un peso del 40% ed il secondo del 60%. La temperatura risultante sarà il valore della prima misura moltiplicato per 0,4 più il valore della seconda moltiplicata per 0.6. Se il primo sensore dovesse guastarsi, il suo peso sarebbe posto a 0 ed il suo valore sarà ignorato. La temperatura d'ingresso sarà derivata direttamente dal secondo sensore. Esso è stato effettivamente disconnesso riducendo a zero il suo peso. Qualora il primo sensore dovesse ancora fornire valori ragionevoli, esso sarebbe «riconnesso automaticamente» semplicemente ri-introducendo un peso non 0 come, per esempio, ancora 0,4.

Nel caso del controllore del traffico, un altro esempio di ingresso non ragionevole potrebbe essere un'indicazione di velocità continua di 0 km/h da uno dei rilevatori ad anello. Questo potrebbe verificarsi a causa di un malfunzionamento reale oppure perchè un'auto è parcheggiata direttamente sopra il rivelatore ad anello. Il vantaggio di questo meccanismo di connessione - sconnessione automatica verrà mostrato di seguito. Il processore determinerà che l'anello sta fornendo un valore non ragionevole poichè tutti gli altri anelli stanno indicando una velocità di 38 km/h. L'anello difettoso sarà automaticamente sconnesso e verrà generato un diagnostico. Quindici minuti più tardi, il microprocessore, che conserverà il controllo di questo anello «difettoso» anche senza utilizzare il suo risultato, noterà che la velocità indicata dall'anello è nuovamente 38 km/h, cioè una velocità ragionevole. Dopo un periodo di diversi minuti di comportamento ragionevole dell'anello, il microprocessore, lo riconnetterà al sistema. Questo incidente si è verificato a causa di un'auto parcheggiata sopra il rilevatore ad anello e poi rimossa. L'anello viene automaticamente riconnesso al sistema in questo caso, mentre esso potrebbe essere «perso» in un sistema «meno intelligente».

Queste tecniche sono state utilizzate per molto tempo nel controllo dei processi industriali che potevano permettersi l'impiego di un minicomputer. Il punto essenziale da ricordare è che queste tecniche devono ora essere utilizzate nel corso di sistemi microprocessori. Esse forniscono un alto livello di capacità ed affidamento del sistema, rese possibili dal processore. Essenzialmente questi meccanismi filtrano malfunzionamenti transitori e consentono il miglior funzionamento del sistema con le risorse disponibili.

Filtraggio programmato

Ogni volta che un sistema microprocessore campiona un grande numero di ingressi su un certo periodo di tempo, l'ingresso deve essere filtrato in modo da eliminare indicazioni spurie e da ottenere il risultato più preciso possibile. Il filtraggio può essere facilmente realizzato per via software. Per esempio, in un multimetro digitale (DMM) la tensione da misurare sarà campionata diverse centinaia di volte al secondo. Si assumerà qui che essa sia campionata 50.000 volte al secondo. La tecnica di filtraggio più semplice possibile ed una delle più efficaci è semplicemente la tecnica della media: 50.000 misure vengono sommate assieme e quindi divise per 50.000. Il valore risultante è il potenziale d'ingresso *mediato* e filtrato. Naturalmente si assumerà che qualsiasi valore non ragionevole venga scaricato. La misura risultante del potenziale avrà un'elevata precisione. Abbiamo così realizzato un filtro programmato.

CONVERSIONE DA ANALOGICO A DIGITALE

Fino a poco tempo fa, la conversione da analogico a digitale era un problema significativo in quanto coinvolgeva un grande numero di componenti rispetto ai pochi necessari per realizzare il sistema microprocessore stesso. Questa situazione è

cambiata. Gli ADC sono ora disponibili su chip singolo a basso costo. Ora i chips ADC comprendono i drivers tri-state ed i buffers richiesti per interfacciare direttamente il bus dati di un microprocessore standard. La realizzazione di interfacce analogiche multicanale può essere fatta con pochi componenti. Inoltre sono commercialmente disponibili moduli e schede che interfacciano direttamente la maggior parte di sistemi microprocessori esistenti. In breve un sistema a microprocessore può ora essere facilmente accoppiato ad un assemblatore di conversione multicanale da analogico a digitale e da digitale ad analogico.

Le due tecniche principali nel campo degli ADC sono le approssimazioni successive e l'integrazione a doppia pendenza. Nella tecnica delle approssimazioni successive un valore d'ingresso analogico è confrontato al valore di un contatore (convertito in analogico) generando un'approssimazione progressivamente più accurata. La precisione risultante è media (da 8 a 10 bit) ma la velocità di conversione è elevata (20 µsec). Esiste ora un grande numero di ADC su chip singolo disponibili nella tecnica delle approssimazioni successive ad un basso costo ($ 10).

L'integrazione a doppia pendenza impiega un condensatore che viene inizialmente caricato durante un numero noto di impulsi di temporizzazione mentre è connesso ad una tensione di riferimento e quindi scaricato al potenziale da misurare. Il tempo trascorso durante la scarica è proporzionale al valore del potenziale. Questa è una tecnica caratterizzata da un'elevata precisione (12 o 13 bit); comunque essa è intrinsecamente lenta (20 msec.). Diversi convertitori su chip singolo sono anche disponibili per la tecnica di integrazione a doppia pendenza ma il costo è talvolta più elevato di quelle per le approssimazioni successive. Il lettore interessato all'interfacciamento specifico di questi chips, come pure alla realizzazione di tecniche generali di conversione da analogico a digitale e da digitale ad analogico è rimandato al nostro libro sulle Tecniche di Interfacciamento.

Come esempio di un microprocessore analogico diretto, il sistema utilizzato per l'impianto di accensione sulla Toronado (General Motors) sarà descritto in «ESEMPI DI STUDIO» nel corso di questo capitolo. Esso utilizza un microprocessore custom prodotto dalla Rockwell.

APPLICAZIONI CONSUMER

Le applicazioni consumer sono caratterizzate da un grande volume e da un costo più basso possibile. Questa è l'area di applicazione dei microcomputer formati da 1 a 2 chips. Altre architetture di microprocessori sono semplicemente non adatte alle applicazioni tipo consumer poichè esse richiedono significativamente più hardware. Per esempio un microcomputer da 1 chip è sufficiente per funzioni di controllo semplici come quelle richieste da una macchina di lavaggio o da un controllore di un forno a microonde. I vantaggi tecnici sono ovvi: il microprocessore elimina l'elettromeccanica oppure la logica cablata. Esso fornisce più funzioni. Esso fornisce un testing di ragionevolezza: per esempio l'utente di una macchina di la-

vaggio potrebbe richiedere un ciclo caldo per tessuti delicati, il microprocessore può lampeggiare un segnale di attenzione oppure non eseguire il comando sbagliato. I microprocessori sono utilizzati per fornire «intelligenza» aggiunta a tali dispositivi consumer. Per esempio il microprocessore F8 è stato successivamente utilizzato in Germania in televisori a colori avanzati per fornire la sintonizzazione e la «programmazione» automatica dell'insieme fino ad un anno avanti. Naturalmente una volta che un microprocessore è installato in un televisore esso costituisce una tentazione per fornire un orologio digitale «gratuito» come pure altre funzioni di programma. Un'altra tentazione è quella di fornire giochi incorporati impiegando lo stesso od un altro microprocessore. Questo è stato realizzato nello stesso tempo sia con lo F8 che con altri microprocessori (riferimento: GRUNDIG televisori a colori).

Il fatto che questo mercato comprende centinaia di migliaia di unità costituisce un forte incentivo per i costruttori. Con un mercato di diverse centinaia di migliaia di unità vale la pena sviluppare qualunque chip LSI. Comunque, a causa del fatto che questo mercato è così allettante, ci si può aspettare che la competizione sarà aspra. Gli acquirenti di tali microcomputer ad 1 chip sono generalmente grandi e ben finanziate compagnie, che acquisteranno componenti al prezzo più basso possibile. Ne risulta che i costruttori possono avere abbassato irrealisticamente i prezzi e successivamente sviluppate le funzioni richieste. Essi possono realizzare un profitto non immediato su tali prodotti. Comunque, sul lungo periodo, i principali fornitori potrebbero realizzare un profitto significativo su delle grosse vendite.

Naturalmente alcuni problemi tecnici si incontrano quando i microprocessori diventano merci di consumo. Per esempio molti dei parametri forniti devono essere memorizzati in una memoria di lettura/scrittura. Usando il semplice esempio della macchina di lavaggio, il sistema microprocessore deve «ricordare» il tipo di tessuto che sta lavorando e le indicazioni fornite dall'utente. Nella macchina tradizionale una manopola rotante mantiene meccanicamente la traccia di dove si trova il ciclo. L'utente poteva disconnettere la macchina accidentalmente o per mezzo di un'interruzione dell'alimentazione. È imperativo che, quando questo succede, la macchina recuperi il più uniformemente possibile. Un cliente potrebbe non essere molto soddisfatto se dopo la sconnessione della macchina dall'uscita e, dopo reinnesto della spina, non si ha più il ricordo di dove è avvenuta l'interruzione. Ne risulta che occorre fornire una certa quantità di memoria non volatile. Questo è dispendioso rispetto al costo del microprocessore stesso. Sono disponibili diverse soluzioni tra le quali ricordiamo: potrebbero essere usate le EAROM oppure una memoria CMOS assistita da una piccola batteria, oppure un dispositivo meccanico. Gli orologi digitali ed i calcolatori tascabili hanno dimostrato che questo problema tecnico può essere risolto nel caso di progetti speciali. Comunque attualmente, non esiste soluzione generale, a basso costo, industrialmente disponibile sul mercato, a questo problema. Questo rappresenta ancora un piccolo ostacolo.

Esempi tipici di merci consumer equipaggiate di microprocessore sono le mac

chine di lavaggio, le macchine per cucire (la Singer Athena 2000 è equipaggiata con un microprocessore custom prodotto dalla AMI), forni a microonde, televisioni a colori. Nel settore dell'ufficio, la maggior parte delle dispendiose macchine per l'ufficio presumibilmente diventerà dotata di microprocessore nel prossimo futuro: macchine da scrivere ad elaborazione di parola, macchine per le fotocopie, centraline telefoniche.

A causa dell'enorme nuovo mercato aperto dai produttori di dispositivi a semiconduttori dalle applicazioni di tipo consumer, cosa che non è mai avvenuta prima dell'avvento dei calcolatori tascabili, verranno introdotti molti nuovi prodotti. Verranno utilizzati sia microprocessori standard che *progetti custom*.

Per le grandi quantità coinvolte è perfettamente ragionevole considerare l'impiego dei progetti custom che offrono due vantaggi: il primo vantaggio è di renderli più immuni alla copiatura da parte della concorrenza: il secondo vantaggio è di renderli più adatti, cioè più a buon mercato per una data applicazione. Un progetto custom ha anche due svantaggi essenziali: primo è dedicato all'applicazione e normalmente sarà molto più difficile riprogrammarlo per un nuovo miglioramento che si desidera introdurre; secondariamente, a causa del progetto per scopi speciali il suo tempo di collaudo potrebbe essere lungo finché esso diviene veramente affidabile.

Per le ragioni precedenti, i progetti custom sono una minoranza. Una volta che il mercato si è solidamente stabilizzato, i progetti custom serviranno per allargarlo. Comunque ci si può aspettare che i microprocessori standard saranno utilizzati per affermare tutte le nuove aree di mercato. Uno di tali nuovi mercati sono i giochi elettronici:

I primi giochi realizzati sugli schermi televisivi sono stati realizzati con la logica standard. I microprocessori a basso costo hanno reso particolarmente attrattiva la realizzazione mediante sistema a microprocessore per consentire una grande varietà di giochi programmati realizzati sullo schermo. Diversi costruttori hanno percorso questa strada e realizzato giochi. Questo stabilisce un nuovo mercato. Improvvisamente questi giochi elettronici hanno acquistato una larga diffusione. L'ovvio risultato è stato che altri costruttori, come la General Instrument, si sono dedicati alla realizzazione diretta dei giochi su chip singolo («chips gioco»). Questi nuovi chips-custom una volta introdotti hanno eliminato dal mercato i principali progetti utilizzanti i sistemi a microprocessori in quanto possono realizzare la stessa funzione ad un costo molto più basso. Questo è il rischio che si corre quando la tecnologia progredisce. Ne risulta che l'introduzione di questi giochi su chip singolo a basso costo, ha rivoluzionato il mercato. La larga diffusione di questi giochi ha condotto ad una raffinatezza del mercato. I costruttori devono ora competere non solo in costo ma anche in complessità e raffinatezza dei giochi che offrono. Conseguentemente i nuovi giochi complessi che vengono ora richiesti non possono essere realizzati su un chip singolo. I nuovi giochi più riusciti utilizzano quindi ancora un

sistema a microprocessore.

Poiché, dal 1977, è possibile realizzare un microcomputer completo su uno o due chips, è probabile che questa soluzione sarà valida per un periodo significativo di tempo. L'impiego di microcomputer standard su 1 o 2 chips consente una facile programmazione di nuovi giochi. A causa del basso costo di tali microcomputer in grosse quantità, ci si può aspettare che tali giochi verranno forniti come opzione del costruttore su tutte le televisioni a colori del prossimo futuro. Questo contrassegna l'introduzione di una capacità di calcolo e di una capacità per scopi generali disponibili in casa. Questa è una fase molto significativa. La disponibilità di un microcomputer all'interno del televisore significa che esso può essere utilizzato per altri scopi senza costo addizionale. Esso può essere utilizzato in una prima fase per i giochi, la sintonizzazione fine e la selezione programmata dei programmi televisivi. In una seconda fase esso potrebbe essere usato per l'elaborazione diretta dei dati, utilizzando lo schermo come mezzo per mostrare i dati. A causa della disponibilità di reti TV via cavo, sarebbe anche possibile connettersi ad un computer centrale. Questa possibilità sarà certamente attuabile in futuro.

Quale sarà la prossima applicazione consumer comprendente microprocessori? La risposta generale a questa domanda è semplice. Qualsiasi dispositivo che costi diverse centinaia di dollari è un probabile candidato. Il costo aggiuntivo per incorporare al sistema un microprocessore/microcomputer è piccolo rispetto al costo globale del dispositivo. Introdotto questo sono valutabili e disponibili funzioni extra per l'«intelligenza» aggiuntiva introdotta.

APPLICAZIONI SPECIALIZZATE

Le applicazioni specializzate sono caratterizzate da alcuni limiti speciali e fondamentali. Normalmente questo limite è la microminiaturizzazione del prodotto. Le aree principali caratterizzate da questo limite sono le applicazioni governative (militari, avioniche, aerospaziali) e medicali. Queste sono caratterizzate da una richiesta assoluta di basso volume, normalmente basso consumo di potenza e spesso disinteresse per il costo software. In tali applicazioni l'uso di un microprocessore potrebbe spesso non essere giustificato per ragioni funzionali o economiche. Esso diviene invece una necessità per il fattore limite sulle ridotte dimensioni.

Le applicazioni «governative» (comprendendo quelle militari-avioniche-aerospaziali), sono state infatti una ragione essenziale per lo sviluppo della tecnologia LSI. Il programma aerospaziale ha fornito infatti gli investimenti necessari per questo sforzo. La tecnologia CMOS, caratterizzata da un consumo di potenza molto basso e da un'elevata immunità al rumore, è stata sviluppata dalla RCA specificamente per le applicazioni avioniche. Inizialmente, in queste applicazioni aerospaziali o militari sono stati utilizzati progetti per scopi speciali. Comunque si è verificato un fenomeno interessante. Quando i microprocessori sono entrati nel mercato di massa l'incentivo finanziario principale per i costruttori era quello di alimentare

questo mercato di massa con nuovi prodotti avanzati, piuttosto che limitarsi ad un mercato governativo che si stava restringendo. Ne è risultata una competizione da parte dei produttori per l'introduzione prima possibile di nuovi dispositivi molto complessi. Ne è conseguito che la complessità di qualsiasi progetto per scopi speciali, che potrebbe essere realizzato per applicazione militare, è divenuta molto più bassa di un progetto commerciale. Recentemente è stato mostrato da diversi studi che i militari utilizzano microprocessori commerciali piuttosto che progetti per scopi speciali. I microprocessori commerciali forniscono semplicemente più funzioni e possono essere più affidabili di qualsiasi microprocessore per scopi speciali. Comunque la standardizzazione è lontana dalla completezza. Per esempio, corre voce che l'aviazione per il nuovo bombardiere F16 usa più di 30 sistemi a microprocessore, di cui più di dieci sono diversi!

I microprocessori sono utilizzati su scheda, per una varietà di funzioni: essi regolano i meccanismi di controllo, dall'iniezione di carburante nei reattori al sistema automatico del controllo di volo. Nei sistemi radar sono utilizzati bit-slices più normali microprocessori monolitici. I bit-slices sono particolarmente utilizzati per applicazioni avioniche. Uno dei problemi principali incontrati nella realizzazione di sistemi di guida radar efficiente è naturalmente, il volume e la massa di tali forniture. Questo problema ora è stato risolto con la disponibilità dei dispositivi bit-slice. Ne risulta che la maggior parte degli algoritmi utilizzati per l'elaborazione radar sono ora realizzati mediante programma piuttosto che mediante cablaggio. Un vantaggio secondario dell'impiego degli slices è stata l'introduzione della flessibilità di programmazione. Questi radar possono essere riconfigurati dinamicamente. Questo è il maggior vantaggio per gli ECM (misuratori-contatori elettronici) impiegati nella EW (electronic warfare: guerra elettronica). I dispositivi bit-slice offrono la possibilità di una riconfigurazione dinamica per rispondere ad una minaccia nuovamente identificata. Per esempio il radar frontale dell'F16 (che è realizzato dalla Westinghouse, impiegando la tecnologia bit-slice) ha ottenuto un guadagno di peso e spazio maggiore del 30%. Il guadagno di spazio ha consentito la realizzazione di funzioni addizionali ECM all'interno del muso dell'aereo. Inoltre il radar fornisce possibilità di funzionamento molto migliorate a causa della sofisticazione degli algoritmi che possono essere realizzati su di esso e poi ridefiniti.

Le applicazioni mediche portatili sono normalmente caratterizzate dagli stessi limiti. Esse devono essere portatili, cioè avere basso peso e volume. Naturalmente esse dovrebbero avere il più basso costo possibile ma questo non è il limite principale. Due tipi fondamentali di applicazioni, utilizzanti i microprocessori, sono state sviluppate. Si possono distinguere sistemi impiantati su un portatore umano e sistemi esterni.

Alcune applicazioni, comprendenti l'impiego di microprocessori su un portatore umano, sono state sviluppate recentemente. Per esempio, gli ordinari pacemakers forniscono la stimolazione cardiaca a fissati intervalli di tempo. Lo svantaggio di questi pacemakers, è che il paziente non può esercitare nessuno sforzo violento. In-

fatti sotto sforzo le tossine emesse dai muscoli nel flusso sanguigno dovrebbero essere eliminate mediante pompaggio più rapido del sangue. Sfortunatamente un pacemaker standard non aumenta il flusso del sangue. Esso mantiene la stimolazione del cuore a velocità costante. I nuovi pacemakers dotati di microprocessore forniscono la stimolazione cardiaca proporzionale al ritmo respiratorio. Sono stati sviluppati semplici sensori che possono fornire quest'ingresso al pacemaker «proporzionale». Sono stati sviluppati altri dispositivi sperimentali che forniscono uno stimolo programmato al sistema nervoso in risposta alla rivelazione di un'anormale attività cerebrale. Chiaramente tali dispositivi sono attualmente solo in fase sperimentale ma non pongono nessun problema tecnico significativo. Sono state proposte numerose altre applicazioni da sviluppare. Una di queste è un dispositivo tipo grande orologio per mostrare il ritmo cardiaco dal rilievo della pressione del sangue al polso. Tale dispositivo ha la capacità di emettere un allarme sonoro ogni volta si può prevedere un malfunzionamento cardiaco. In alcune malattie o malformazioni cardiache, un attacco cardiaco può essere previsto con un elevato livello di accuratezza, diversi minuti o più prima che esso effettivamente si verifichi. Un tale dispositivo potrebbe quindi fornire il tempo sufficiente per un'azione preventiva dove possibile.

I sistemi esterni impieganti microprocessori nelle applicazioni medicali sono essenzialmente analoghi ai sistemi di controllo industriali. Essi sono normalmente asserviti ad una funzione di osservazione di processo. Offrono il vantaggio di velocità, affidabilità ed intelligenza. Per esempio la maggior parte delle funzioni vitali di un paziente possono essere osservate direttamente da un sistema di elaborazione posto di fianco al letto. Questo fornisce l'osservazione 24 ore su 24 del ritmo cardiaco, della pressione del sangue e di altre funzioni necessarie. Ogni volta che si rileva un fenomeno anomalo può essere possibile prevedere un problema medico utilizzando la potenza del calcolo di un processore. Inoltre l'utilizzazione di un tale sistema automatico origina un'affidabilità più elevata di quella dell'impiego del personale medico per tali funzioni di osservazione, specialmente di notte.

I microprocessori possono anche essere utilizzati per i diagnostici o per le operazioni medicalmente correlate automatizzabili ogni volta che esiste un chiaro algoritmo. Essi sono utilizzati per il conteggio delle celle del sangue e per altre prove di laboratorio dove sono essenziali velocità, precisione ed affidabilità. Si potrebbero considerare molte altre applicazioni che saranno realizzate in un prossimo futuro. I microprocessori possono essere vantaggiosamente utilizzati per la raccolta dati sia nell'ambiente commerciale di un ospedale oppure in un laboratorio, come pure per scopi medici. Inoltre ci si può aspettare che gli strumenti equipaggiati di microprocessori saranno sviluppati per l'uso dei fisici nei loro laboratori originando una maggiore accuratezza nella rivelazione di fenomeni vitali anomali.

ALTRE AREE DI APPLICAZIONE

Sono state tracciate le differenze tra le quattro aree di applicazione precedente-

mente indicate. Ci si può aspettare che queste differenze diminuiscano rapidamente. All'allargarsi dell'impiego dei microprocessori, tutti i tipi di microprocessori saranno utilizzati virtualmente per ogni tipo di applicazione. È ancora interessante differenziare i tipi di applicazione visto che così richiedono oggi diversi tipi di forniture. Le linee indicate si uniranno in futuro. Quindi non si farà nessun ulteriore sforzo per la differenziazione dei possibili tipi di applicazioni. Ci si concentrerà quindi sulla realizzazione di applicazioni effettive.

REALIZZAZIONE DI UN'APPLICAZIONE DEI MICROPROCESSORI

Abbiamo mostrato nei precedenti capitoli come è semplice assemblare una scheda CPU. L'assemblaggio di un sistema per una data applicazione è pressochè altrettanto semplice. I due problemi principali che devono essere risolti specificamente per ogni applicazione sono l'interfacciamento e la programmazione. Questi due problemi saranno sviluppati nei due successivi capitoli. Dal 1977 le interfacce più comuni richieste per connettere i dispositivi d'ingresso-uscita standard sono disponibili nel formato chip singolo e possono essere connessi semplicemente al sistema.

Useremo un microprocessore standard (in questo caso un dispositivo a 4 bit, il 4040) per realizzare applicazioni progressivamente più complesse. Si vedrà che l'architettura del sistema rimane costante. Ulteriori funzioni si ottengono connettendo più moduli hardware. L'assemblaggio di qualunque dei sistemi che saranno descritti utilizzando altri microprocessori come il 6800, il 2650 e l'8080 è del tutto simile. Costruiremo inizialmente un dispositivo d'ingresso di base con un ingresso esadecimale ed un'uscita LED e quindi un controllore del lettore-perforatore del nastro di carta, un controllore di cassette ed infine l'aggiunta di caratteristiche analogiche per il controllo industriale diretto.

UN CONTROLLORE DI PANNELLO FRONTALE

Sono richiesti solo quattro chips LSI (vedere Fig. 6.1): la MPU 4040 più il suo clock, il 4201. Una RAM 4002 fornisce le capacità I/O di lettera/scrittura. Una ROM 4308 - I/O consente la memorizzazione del programma e le capacità I/O. L'incentivo nel progetto che si sta descrivendo è la riduzione del numero di componenti. A questo fine si useranno chips speciali che comprendono sia memoria che capacità I/O. In un sistema più grande dove il limite costo potrebbe non essere essenziale i chips di memoria ed i chips di I/O probabilmente sarebbero separati.

I due chips speciali appena introdotti, il 4002 ed il 4308, forniscono sia capacità di memoria che caratteristiche I/O. Ciascuno fornisce 16 linee di I/O. L'impiego di queste linee sarà illustrato nelle applicazioni.

Il controllore di pannello frontale è inteso a fornire caratteristiche d'ingresso attraverso la tastiera e caratteristiche di display attraverso il display a quattro digit e può essere in grado di comunicare con un computer esterno attraverso un bus a

16 bit. Il programma è contenuto nella ROM 4308. L'area scratch-pad per la memorizzazione temporanea dei dati e per l'esecuzione dei calcoli intermedi è fornita dal 4002. Osserviamo in maggiore dettaglio le funzioni d'ingresso-uscita fornite dal 4308:

Quattro pins del 4308 sono utilizzati per la connessione alle colonne dei 16 tasti della tastiera. Viene utilizzata una tecnica di scansione. Questa sarà descritta in dettaglio nel prossimo capitolo dedicato all'interfacciamento. Quattro pins sono utilizzati per raccogliere dati dalle quattro righe della tastiera. Altri quattro pins sono necessari per la conessione al display LED. Per i LED sette segmenti standard

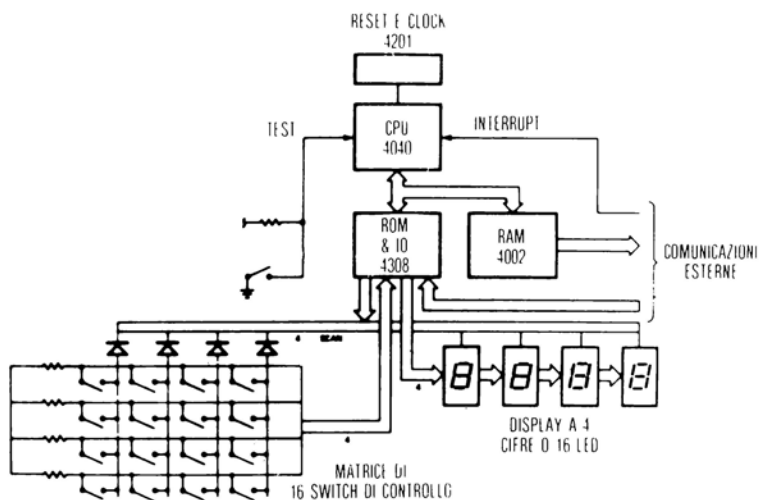



Fig. 6-1: Un controllore di pannello frontale.

potrebbero essere richieste solo tre linee. L'ultimo rimanente dei quattro bit può essere usato per qualunque scopo come la comunicazione con il computer-ospite. La comunicazione con questo computer viene realizzata utilizzando 16 linee uscenti dal 4002 sulla parte destra dell'illustrazione. L'interrupt può essere utilizzato oppure no.

Questo è uno degli esempi più semplici di applicazione possibili. Esso è essenzialmente analogo al calcolatore tascabile più le linee di comunicazione al mondo esterno. Utilizzeremo semplicemente questo sistema di base per la realizzazione di applicazioni progressivamente più complesse. Si aggiungeranno più chips LSI per ottenere le funzioni richieste.

UN CONTROLLORE DI LETTORE-PERFORATORE DI NASTRO DI CARTA

Si realizzerà ora un controllore di lettore-perforatore di nastro di carta. Tale controllore di nastro legge sette oppure otto bit di dati ASCII dal nastro e perfora i dati con il suo meccanismo di perforazione. Inoltre esso è equipaggiato con un pannello frontale (o «pannello di controllo») che richiede otto linee per la comunicazione. Quattro linee sono necessarie per dare ordini al meccanismo e quattro linee di rivelazione sono necessarie per prelevare da esso informazioni sugli interruttori e sulle

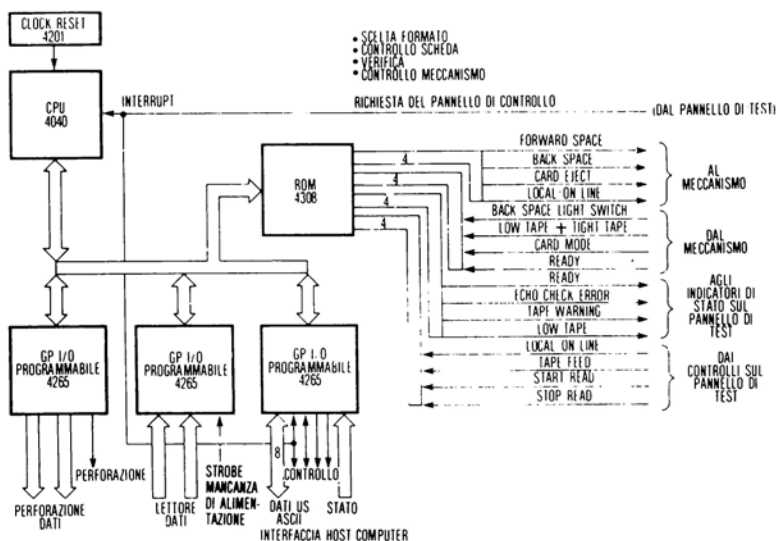



Fig. 6-2: Un controllore di perforatore-lettore di nastro di carta.

indicazioni di stato. Le funzioni richieste saranno ottenute aggiungendo due chips al sistema precedente. L'architettura del sistema appare in Figura 6-2. È stata aggiunta una ROM 4308 che fornisce l'ulteriore memoria di programma richiesta da questa applicazione più complessa e le 16 linee di I/O.

Inoltre viene aggiunto un nuovo chip: il 4211 GPI/O. Il 4211 è un chip di interfaccia, per scopi generali che fornisce 16 linee che sono singolarmente programmabili in direzione. Se non fosse necessaria più memoria di programma, si potrebbe utilizzare un 4211 invece della precedente ROM 4308.

La realizzazione effettiva è riportata in Figura 6-2. Nella parte alta a destra dell'illustrazione una delle ROM 4308 è utilizzata per interfacciare il lettore-perforato-

re del nastro di carta. Otto linee interfacciano al meccanismo ed otto linee interfacciano al suo pannello di controllo, come indicato.

I dati scambiati con il lettore-perforatore del nastro di carta transitano su un bus bidirezionale ad 8 bit che appare in basso nell'illustrazione (dati ASCII).

Infine la comunicazione con il computer ospite attraverso un bus dati bidirezionale a 16 bit appare in basso a sinistra nell'illustrazione ed è eseguita utilizzando la seconda ROM 4308.

La logica di controllo richiesta per interfacciare al meccanismo lettore-perforatore è stata realizzata usando essenzialmente sei chips. Aggiungiamo un altro grado di complicazione:

UN CONTROLLORE DI COMANDO DI CASSETTE

Si realizzerà un controllore di cassette usando solo un chip addizionale rispetto all'esempio precedente. Un ulteriore 4211 GPI/O è utilizzato per fornire le 16 linee di controllo richieste da-al meccanismo del nastro. I dati provenienti da oppure inviati al comando del nastro sono seriali. Si assumerà che questi dati vengano convertiti da seriali a paralleli 4 bit. Questa funzione potrebbe essere eseguita da un'unità UART. La struttura del sistema appare in Fig. 6-3. I dati provenienti dalla cassetta del nastro oppure inviati ad essa arrivano alla ROM 4308 che compare sulla destra dell'illustrazione. Il chip 4211 GPI/O in alto a destra dell'illustrazione fornisce le 16 linee richieste dal meccanismo. Ogni linea è etichettata per indicare la funzione eseguita. Infine le linee rimanenti del 4308 sono utilizzate per fornire le funzioni di controllo necessarie sia per scopi di trasmissione dati che per interfacciare il pannello di controllo ovvero le informazioni di rivelazione provenienti da quest'ultimo. I dati accumulati da questo controllore di cassette saranno trasmessi a oppure da un computer ospite attraverso un bus dati bidirezionale a 16 bit che appare in basso nell'illustrazione. La ROM 4308, la RAM 4002 ed il secondo chip 4211 GPI/O sono utilizzati per questo scopo.

Qualunque ulteriore funzione I/O che potrebbe essere richiesta, si potrebbe ottenere usando un ulteriore chip 4211 GPI/O. Se fosse necessario realizzare un programma più lungo, la memoria richiesta sarebbe fornita usando più chips ROM e più chips 4308.

Gli esempi potrebbero essere resi «complicati» aggiungendo funzioni ed i chips richiesti. Finchè l'unità microprocessore utilizzata per le funzioni di calcolo è abbastanza veloce da fornire il tempo di risposta richiesto per gli algoritmi che sono stati realizzati, la nostra discussione è valida. Se il 4040 utilizzato in questo esempio dovesse dimostrarsi troppo lento, si potrebbe sostituire con un microprocessore più veloce, come un progetto ad 8 bit.

La conclusione essenziale derivante dagli esempi precedenti è la seguente: non esiste essenzialmente il problema del progetto hardware nell'assemblaggio della

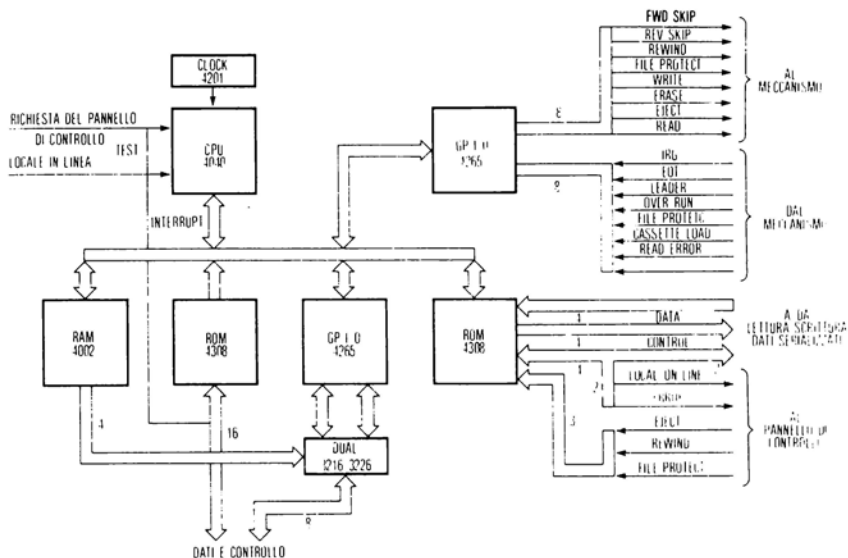
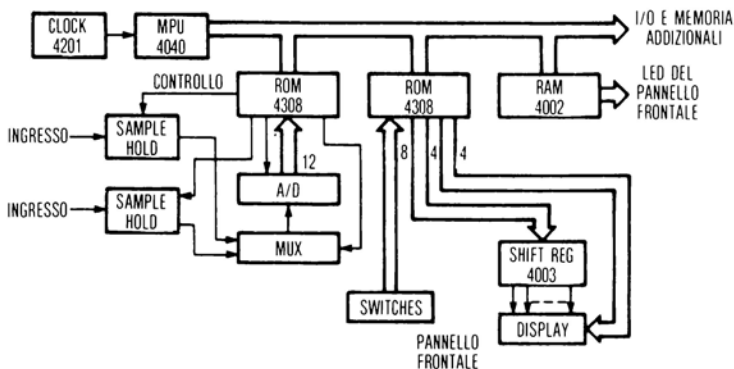
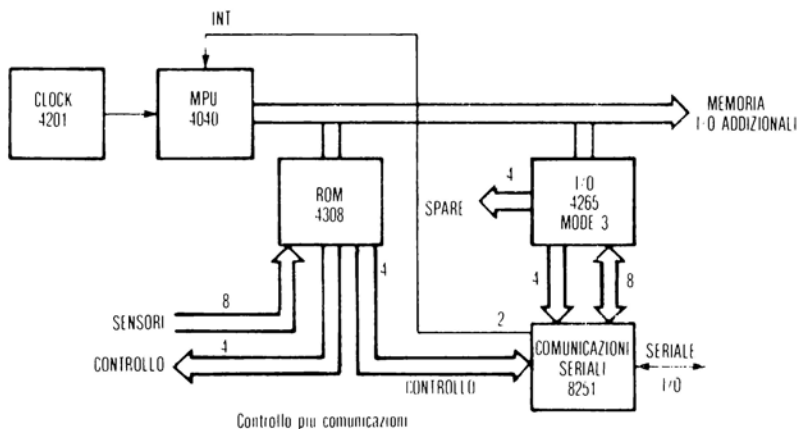



Fig. 6-3: Un controllore di drive di cassette.

scheda di controllo di base. Le funzioni si ottengono semplicemente connettendo i chips ulteriormente richiesti al sistema finchè la velocità di elaborazione è sufficiente per l'esecuzione del lavoro. Naturalmente non abbiamo considerato la complessità addizionale di scrittura del software e questo problema verrà affrontato in seguito.

Un'importante caratteristica addizionale richiesta in molti sistemi è l'ingresso e l'uscita analogici. Questa funzione può essere facilmente ottenuta e verrà esaminata in questa sede:

Controllo I/O a 4-bit con A/D

INGRESSO - USCITA ANALOGICI

È possibile fornire funzioni d'ingresso - uscita analogiche altrettanto facilmente quanto l'ottenimento delle funzioni digitali richieste nell'esempio precedente. Per la conversione da digitale ad analogico si userà un DAC ed uno o più ADC saranno utilizzati per la conversione da analogico a digitale. Le tecniche di conversione sono state precedentemente descritte in questo capitolo. Il nostro semplice sistema 4040 appare in Figura 6-4, configurato in modo tale da fornire caratteristiche analogiche.

Un ADC ad 8 bit (bassa precisione) è utilizzato per convertire un segnale analogico esterno in un valore digitale di 8 bit. Uno o più DAC sono connessi sulla de-

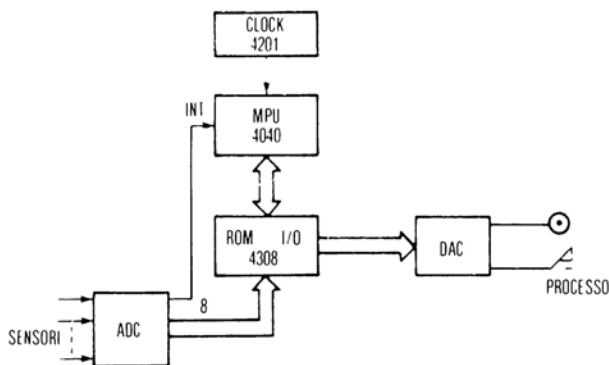



Fig. 6-4: Controllore di processo di base.

stra del sistema per convertire l'uscita digitale in un segnale analogico che è connesso a monitors oppure relé.

In un'effettiva applicazione industriale, l'ADC potrebbe essere generalmente interfacciato ai segnali analogici esterni attraverso un *multiplexer* ed uno o più circuiti *sample and hold* (di campionamento) che congelano l'informazione per l'ADC. Il ruolo del multiplexer è quello di connettere diversi segnali analogici utilizzando solo un convertitore da analogico a digitale (dispendioso), nell'ipotesi che la velocità di conversione risultante sia sufficiente.

Ogni volta che la velocità di conversione non è essenziale sarà utilizzato un multiplexer analogico in modo da ridurre il dispendio dei componenti. Un esempio tipico appare in Figura 6-5. Il multiplexer in basso a destra nell'illustrazione può selezionare uno degli otto ingressi analogici. La selezione viene eseguita specificando un codice di 3 bit. Tre linee del bus indirizzi, indicate A_0, A_1, A_2 sono utilizzate per selezionare i segnali d'ingresso. Il segnale analogico selezionato viene quindi inviato, attraverso un amplificatore, al circuito sample and hold che lo congelerà. Dal microprocessore viene utilizzata una linea per fornire il segnale di congelamento al circuito sample and hold (abbreviato S/H). L'ADC può quindi utilizzare il valore campionato nel circuito S/H e convertirlo in forma digitale. Un ordine di conversione, verrà fornito dal microprocessore su una linea separata che appare nell'illustrazione. L'ADC fornirà quindi otto bit di dati o più al microprocessore. Un ADC a 10 o 12 bit trasmetterà prima gli otto bit d'uscita e quindi i rimanenti bit sul bus dati del microprocessore.

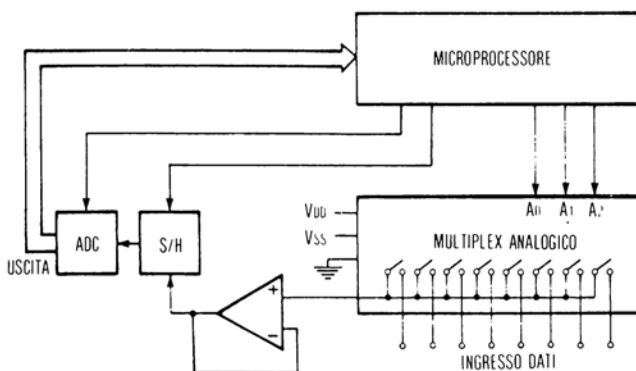



Fig. 6-5: Un sistema di acquisizione dati ad 8 canali.

Il sistema appena descritto è un sistema di acquisizione dati ad 8 canali. L'aggiunta di caratteristiche analogiche al nostro sistema di base è stata ottenuta ancora una volta con un modesto numero di chips addizionali connessi alla normale ar-

chitettura standard. La regola è la seguente: altre funzioni per scopi speciali che possono essere richieste in un'applicazione saranno fornite dalla connessione dell'interfaccia richiesta allo stesso sistema microprocessore di base. L'architettura del sistema è essenzialmente congelata. Ne risulta che qualsiasi scheda standard di microprocessore può essere utilizzata nella maggior parte delle applicazioni. Il solo compromesso tecnico in una data applicazione può essere presente nell'area di interfaccia e sempre nella sua programmazione.

Allo scopo di illustrare alcuni dei limiti attuali e le soluzioni nelle applicazioni reali saranno presentati alcuni esempi di studio.

DOMANDA 1: *Qual è il vantaggio del circuito Sample and Hold (S/H)?*

DOMANDA 2: *Parlate dei vantaggi della precisione di un ADC a 8 bit rispetto ad un ADC a 10 bit.*

DOMANDA 3: *È necessaria la MPU in Figura 6-5?*

ESEMPI DI STUDIO

Di seguito verranno presentati tre esempi di studio:

Un'applicazione industriale comprendente un sistema a microprocessori standard: un controllore del traffico urbano.

Una seconda applicazione industriale comprendente un processore per scopi speciali: un sistema di temporizzazione di scintille per automobile.

Un'applicazione di tipo consumer: un controllore di microonde.

Questi tre esempi serviranno ad illustrare i concetti e le tecniche che sono stati introdotti per spiegare le reali motivazioni dei progetti specifici.

Un controllore del traffico urbano

L'uso dei microprocessori per il controllo semafori rappresenta uno dei primi impieghi pubblici dei microprocessori in un contesto industriale. (L'autore di questo libro ha realizzato uno di tali primi progetti). La regolazione delle luci del traffico agli incroci è stata tradizionalmente ottenuta utilizzando controllori elettromeccanici nei casi semplici e controllori «elettronici» nei casi complessi. Gli algoritmi per il controllo del traffico in una zona urbana sono progressivamente accresciuti in complessità richiedendo caratteristiche elettroniche complesse per la temporizzazione delle luci. Alcune delle funzioni che devono essere eseguite da un controllore delle luci del traffico sono:

- La temporizzazione di base di ogni fase (una fase è un raggruppamento logico delle luci del traffico: rosso - giallo - verde).
- Selezione dei cicli di temporizzazione. Questo può comprendere un certo nu-

mero di cicli di base come la temporizzazione normale, quella notturna, il ciclo delle ore di punta ed altri in dipendenza della durata del giorno oppure dalla misura dei parametri del traffico.

- Sequenza di inizializzazione dopo l'accensione dell'alimentazione.
- Funzioni speciali come la priorità (da parte della polizia o dei veicoli di emergenza).
- Attuazione da parte dei pedoni e delle macchine che attraversano i rivelatori ad anello. (modo «sensibile al traffico»).
- Moduli per il calcolo dei parametri del traffico richiesti come la «densità» oppure il «volume».
- Moduli per la connessione a linee di trasmissione che connettono il controllore ad un'unità centrale oppure ad altri controllori del traffico in una rete. Tipicamente sono richiesti un ricevitore, un trasmettitore ed un modem.

La struttura del controllore del traffico equipaggiato di microprocessore appare in Figura 6-6. I moduli saranno descritti al paragrafo successivo. I limiti principali dei tradizionali controllori elettronici del traffico sono:

- Costo. Qualsiasi cambiamento nelle funzioni richieste è dispendioso in quanto comporta un nuovo progetto e cablaggio estensivo.
- Limiti di complessità degli algoritmi che possono essere realizzati.
- Adattamento richiesto per ogni nuova intersezione di qualsiasi algoritmo diverso.
- Bassa affidabilità del sistema dovuta ad elevata complessità hardware.

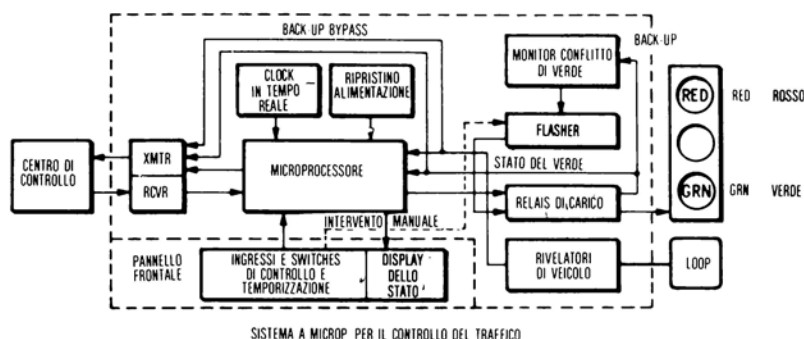



Fig. 6-6: Controllore di traffico urbano.

I minicomputer sono stati utilizzati con buoni risultati per la realizzazione di procedure di controllo molto complesse. Comunque il costo e le dimensioni dei si-

stemi minicomputer è tale da rendere proibitivo il loro impiego nei controllori del traffico convenzionali (il cui costo è tipicamente minore di 3000 dollari).

Il microprocessore ha reso possibile la sostituzione della maggior parte dei complessi moduli hardware mediante equivalenti software. Un controllore del traffico equipaggiato di microprocessore utilizza semplicemente una scheda microprocessore standard più le caratteristiche d'interfaccia richieste. Tutte le funzioni specifiche del sistema sono realizzate mediante programma. Il sistema illustrato in figura 6-6 verrà ora illustrato. Al centro dell'illustrazione appare il modulo microprocessore che rappresenta la CPU del sistema, cioè la scheda basata sul microprocessore che fornisce la memoria, le caratteristiche I/O e CPU. Alla sommità dell'illustrazione appaiono due moduli speciali: essi sono il *clock in tempo reale,* necessario per la precisa temporizzazione degli eventi esterni, e l'unità di *ripristino dell'alimentazione* necessaria per il ripristino del sistema dopo una caduta dell'alimentazione come pure per l'immagazzinamento dei dati essenziali in corrispondenza di una caduta dell'alimentazione.

Le funzioni di rivelazione e di controllo sono realizzate sulla destra dell'illustrazione. Le strategie di controllo del traffico si basano sull'informazione fornita dai rivelatori di veicolo. Il rivelatore di veicolo più spesso utilizzato è il rivelatore ad anello. Un foro rettangolare viene praticato sulla pavimentazione in corrispondenza dell'incrocio. Due o più anelli di conduttore sono depositati all'interno della fenditura, e le estremità dell'anello sono connesse ad un circuito oscillante RC. La frequenza di oscillazione dipende dall'impedenza dell'anello. La presenza di qualsiasi grossa massa magnetica, come un'automobile, su una parte o sull'intero anello cambierà la sua induttanza e la sua frequenza di oscillazione. Questo scorrimento di frequenza sarà rivelato con facilità ed originerà la chiusura di un interruttore oppure di un relè. Questo genererà un segnale discreto che può essere osservato direttamente da un microprocessore. Un blocco *rivelatore di veicolo* che appare sulla destra dell'illustrazione, è l'unità incaricata della conversione dello scorrimento di frequenza in un segnale discreto esterno indicante la presenza di un veicolo sull'anello. Gli anelli saranno normalmente installati in ogni via dell'intersezione. Essi possono essere utilizzati per diversi scopi. Essi possono rivelare la presenza di un veicolo, ma possono anche essere utilizzati per la misura di velocità.

Le misure di velocità possono essere realizzate in due modi. Il principio di base è quello di calcolare la velocità in base alla relazione S=D/T dove D è la distanza e T è il tempo. Il metodo più preciso consiste nell'impiego di due rivelatori ad anello, distanti D metri. Il microprocessore misura semplicemente il tempo T che separa le due successive presenze d'impulsi in ciascuno dei due rivelatori ad anello, e calcola la velocità applicando la semplice formula precedente. Un metodo semplificato alternativo viene utilizzato ogni volta che è disponibile un solo anello: assumendo una dimensione media del veicolo, la durata della presenza d'impulso fatto scattare da un veicolo che attraversa il rilevatore ad anello può essere semplicemente utilizzata per calcolare la stima della velocità.

I rilevatori ad anello sono lo strumento essenziale per fornire il controllo dinamico del traffico (ottimale) agli incroci. Il ruolo del processore è quello di misurare più parametri possibile e di ottimizzare il traffico in funzione di queste indicazioni. Si possono realizzare diverse strategie di ottimizzazione. Può essere desiderabile ottimizzare il numero di veicoli che stanno attraversando un qualsiasi incrocio assegnato. Può essere desiderabile ottimizzare la velocità dei veicoli in un dato incrocio. Può essere desiderabile ottimizzare il flusso di automobili attraverso diverse strade in una rete urbana. Infine, in molti casi può essere desiderabile garantire un'onda «verde» lungo arterie specifiche. Un'onda verde significa che ogniqualvolta un'automobile entra nell'arteria, e supponendo che mantenga una velocità raccomandata, essa percorrerà completamente la strada senza alcun arresto ai semafori. In un'assegnata rete di traffico urbano, può essere utilizzata una combinazione di queste strategie di ottimizzazione. Esse sono mutamente incompatibili e l'ottimizzazione del flusso del traffico su una rete completa si risolve con grande complessità.

Verrà ora esaminata la parte rimanente del meccanismo di controllo. Si osservi ancora la parte destra dell'illustrazione 6-6. Il microprocessore deve comandare l'accensione in sequenza delle luci del traffico (verde, giallo, rosso). A causa della potenza impiegata, la commutazione è realizzata attraverso relè di carico.

È stato sottolineato che in qualsiasi sistema di controllo industriale è imperativo che sia fornita la retroazione di stato per verificare l'esecuzione corretta di un'ordine. Questo è realizzato dalla linea di stato uscente dal relè di carico ed entrante nel microprocessore. Al minimo si dovrebbe osservare lo stato di verde ed idealmente si dovrebbe osservare lo stato di tutte tre le luci. Ogni volta che il microprocessore deve accendere il verde, esso controllerà alcuni millisecondi più tardi che il relè di carico sia stato sicuramente chiuso, verificando il suo stato. Se la risposta è negativa l'ordine verrà inviato altre volte. Se il non funzionamento del relè di carico è permanente, si vedrà che viene messo in funzione un modo di emergenza per un funzionamento sicuro del sistema. Utilizzando l'informazione di retroazione si può garantire che, a meno di malfunzionamenti del microprocessore, non si dovrebbero verificare grossi problemi di controllo. Il sistema può essere ulteriormente raffinato derivando l'informazione di stato direttamente dalle lampade del semaforo. Questo risolve il problema creato da una lampada del verde fulminata.

È stato anche indicato che la caratteristica di *guasto leggero* deve essere fornita nel caso di malfunzionamento hardware o software. Questo è il ruolo del monitor di conflitto del verde e della sua unità lampeggiatrice, che appare in alto a destra nell'illustrazione. Il monitor di conflitto del verde osserva continuamente lo stato del verde per tutte le lampade dell'incrocio. Se due verdi in conflitto dovessero essere accesi contemporaneamente, il monitor di conflitto del verde rivelerà questo evento e disconnetterà automaticamente il sistema microprocessore (assumendo che esso sia in avaria) ed accenderà l'unità lampeggiante. Detta unità lampeggerà alternativamente rosso e giallo nella direzione coinvolta. Questo degrada il funzionamento del sistema ma evita una situazione potenzialmente disastrosa avente le luci

verdi abilitate in direzioni che risulterebbero in collisione. Questo sistema è impiegato in California e nella maggior parte degli stati degli U.S.A. Se si verificasse un malfunzionamento software o hardware risultante in conflitto di verde, questo sistema rivelerebbe automaticamente la condizione ed eseguirebbe l'azione indicata. Questo è chiamato un meccanismo di guasto leggero. Il sistema completo non viene completamente disabilitato, ma soltanto una parte delle sue funzioni sono disabilitate dal malfunzionamento. Naturalmente, all'interno del microprocessore stesso sono utilizzate tecniche di guasto leggero molto più raffinate per diagnosticare o correggere un certo numero di altre possibili condizioni di errore ai livelli d'ingresso o d'uscita.

Due linee contrassegnate «bypass di ritorno» appaiono nella parte alta dell'illustrazione. Esse connettono il trasmettitore ed il ricevitore sulla sinistra del sistema microprocessore. Queste linee sono rispettivamente l'informazione del rivelatore ad anello e lo stato del verde. Esse sono trasmesse ad un centro di controllo. In pratica in città è installata una rete di tali sistemi di controllo del traffico mediante microprocessore. Ciascuno di questi comunica con un centro di controllo del traffico che può essere equipaggiato con un minicomputer di coordinazione. L'informazione mostrata nel centro di controllo comprende lo stato di ciascun incrocio (soltanto il verde oppure verde-giallo-rosso). Inoltre, al centro di controllo la densità del traffico può essere semplicemente valutata mostrando le attuazioni provenienti dai rivelatori ad anello. Naturalmente, quest'informazione può anche essere misurata al centro di controllo e poi codificata e mostrata in forma digitale. Quest'informazione essenziale d'ingresso e retroazione di stato può anche essere inviata ad un controllore del traffico contiguo per la sincronizzazione dei controllori successivi lungo un'arteria. Se si dovesse verificare un malfunzionamento, un secondo microprocessore potrebbe concepibilmente assumere la funzione del primo se esso era stato equipaggiato con una linea addizionale che lo collega ai relè di carico. Nel caso di un significativo flusso d'informazione trasmessa tra il controllore locale del traffico ed il centro di controllo, viene utilizzato il multiplexing a divisione di tempo (TDM: time division multiplex) per la codifica dei dati su una singola linea di comunicazione. Può essere utilizzato un microprocessore per fornire le caratteristiche TDM in software, richiedendo soltanto un'unità hardware TDM separata.

Infine, dall'osservazione della parte bassa dell'illustrazione, il pannello frontale fornisce la richiesta interfaccia umana locale. Il pannello frontale comprende interruttori di controllo, gli elementi di temporizzazione e l'informazione del display. Gli interruttori di controllo sono utilizzati dall'ingegnere del traffico per specificare localmente un certo numero di parametri. In particolare, gli intervalli di temporizzazione per il giallo sono normalmente specificati da interruttori rotanti o di altro tipo, posizionati localmente. Inoltre la selezione tra i vari modi di funzionamento, normalmente può essere specificata direttamente da interruttori posizionati localmente. Lo stato del sistema, comprendente il display delle luci del traffico su piccoli LED, è normalmente fornito direttamente su questo pannello frontale («display del-

lo stato»). Inoltre il sistema deve essere equipaggiato di possibilità di funzionamento normale oppure di caratteristiche di comando normale per il funzionamento di emergenza. Queste caratteristiche, per esempio, possono essere utilizzate dalla polizia quando si verifica un incidente. In questo caso la regolazione dell'incrocio può essere eseguita in un modo specifico oppure attraverso pulsanti manuali sequenziatori posizionati sul pannello frontale. La linea di funzionamento manuale fornisce questa possibilità e connette il pannello frontale al lampeggiatore, come indicato nell'illustrazione. Questo consente al personale autorizzato il posizionamento manuale del modo lampeggiante alternativo del rosso e del giallo.

Non si è ancora descritto l'insieme dei moduli funzionali del sistema. Brevemente, verrà ora discussa la loro influenza sul costo. I sistemi di comunicazione implicano almeno un costo di diverse centinaia di dollari. Il monitor di conflitto del verde ed il lampeggiatore rappresentano pure almeno diverse centinaia di dollari. Analogamente i relè di carico ed i rivelatori ad anello contribuiscono almeno per centinaia di dollari. Il pannello frontale implica pure un costo simile, senza contare il contenitore metallico e l'installazione.

Conseguentemente il costo della scheda microprocessore è forse il costo più piccolo di qualsiasi altro blocco del sistema!

Quali sono i vantaggi derivanti dall'impiego del microprocessore? In dipendenza del costo elevato dei vari blocchi richiesti in tale sistema, l'introduzione del microprocessore al posto della logica cablata non rappresenta un risparmio significativo nella produzione di un piccolo numero di tali unità. Il principale vantaggio non finanziario dell'impiego del microprocessore è la disponibilità illimitata di *intelligenza* che conduce alla realizzazione degli algoritmi di controllo. Il suo valore più significativo è la rimozione dei limiti della logica cablata per quanto riguarda la complessità degli algoritmi che potrebbero essere ragionevolmente realizzati. L'aspetto del costo e dell'intelligenza non verrà considerato ulteriormente.

Esiste un costo nascosto importante in un controllore del traffico: ciascun controllore del traffico normalmente deve essere fatto su misura per un certo incrocio. La geometria dell'incrocio stesso, il numero effettivo di fasi delle luci del traffico e, fondamentalmente, gli algoritmi o la combinazione di algoritmi che devono essere realizzati localmente costituiscono i parametri più comuni per fare su misura dell'incrocio un controllore del traffico. Quindi occorre inserire diversi moduli hardware in un progetto tradizionale; tra questi contatori di volume di traffico, moduli di sequenzializzazione del tempo nell'arco del giorno, oppure moduli di attuazione pedonali. Il vantaggio essenziale del microprocessore è la sostituzione di questi moduli hardware mediante programmi o sottoprogrammi software. Le combinazioni richieste di programmi software per realizzare le funzioni specifiche possono essere assemblate facilmente sia mediante scelta manuale, sia automaticamente. Essi saranno «bruciati su PROM» che verrà inserita su un sistema di controllo standard di un incrocio. Per fare su misura il controllore all'incrocio è possibile agire esclusiva-

mente a livello software (eccetto per le ovvie scelte del numero di rivelatori, ad anello e del numero di relè di carico) e quindi è possibile produrre in massa controllori del traffico identici e standardizzati. Tutti gli adattamenti richiesti e la programmazione su misura sono eseguiti da un potente sistema di sviluppo a livello software. Inoltre, a causa della possibile produzione di massa di unità hardware identiche, si ha un decremento significativo del costo hardware del controllore globale. Un altro vantaggio per l'utente è che una volta che il sistema è installato il suo funzionamento o gli algoritmi possono essere cambiati semplicemente inserendo nuove PROM. Le caratteristiche software come l'osservazione dello stato di monitoraggio danno un'affidabilità molto maggiore e la rivelazione automatica di guasti dei relè di carico oppure delle lampade dei semafori.

La disponibilità di logica programmata all'incrocio ha consentito la realizzazione di una grande varietà di nuovi e complessi algoritmi. Per esempio è ora possibile la realizzazione di complesse *onde verdi*. Per un certo numero di arterie più importanti di una rete è garantito che si può procedere attraverso tutta o la maggior parte della città senza mai arrestarsi. Questo è realizzato dalla trasmissione di informazioni tra due microprocessori del traffico vicini in una rete e/o coordinando il funzionamento della rete da una centrale. Inoltre una varietà di algoritmi alternativi può essere fornita contemporaneamente su un singolo controllore. Il costo degli algoritmi addizionali forniti è semplicemente il costo dei chips PROM addizionali e non più quello dei costosi moduli hardware addizionali.

Tipicamente un controllore del traffico equipaggiato di microprocessore funzionerà in uno dei tre modi seguenti:

- Il controllore parte sempre dal modo 0 ovvero «modo restart» dove si assume che non sia disponibile nessuna informazione. Questo è un modo di alimentazione che deve essere usato finché divengano disponibili gli altri parametri del sistema. I due parametri di sistema fondamentali di un controllore sono: a) il tempo del giorno e b) le misure effettive del traffico.

- Il modo successivo di funzionamento è il «tempo del giorno». Il microprocessore realizzerà uno fra diversi programmi di temporizzazione, in dipendenza del valore del clock. Questi «programmi di timing» sono stati sviluppati da un engineer del traffico locale e sono utilizzati durante i segmenti chiave del giorno. Il funzionamento tipico del modo «tempo del giorno» comprende temporizzazioni speciali per l'ora di punta e per il funzionamento notturno.

- Il mondo successivo di funzionamento è «attuazione parametri». Dopo un certo tempo di funzionamento del microprocessore, i parametri del traffico possono essere determinati, come la velocità dei veicoli la densità e la distanza, tra i veicoli. In funzione di questi parametri si possono realizzare strategie di controllo del traffico più sofisticate. Se il sistema è dotato di memoria sufficiente, esso può anche utilizzare dati che sono stati rilevati durante i giorni o

le settimane precedenti e confrontarli con quelli attuali.

Infine è ora possibile considerare i sistemi auto-ottimizzanti completamente dinamici in modo approfondito. Un microprocessore partirà dal modo tempo del giorno, passando poi al funzionamento attuazione parametri non appena diviene disponibile un numero sufficiente di parametri, e quindi, possibilmente, in un modo auto-ottimizzante. Sfortunatamente il controllo del traffico è molto complesso e devono essere fatte le misure contemporaneamente di un insieme di parametri. Non esiste una formula matematica semplice che consenta l'ottimizzazione diretta della temporizzazione della rete. Devono essere utilizzate le tecniche euristiche. In ogni caso è possibile sperimentare nuove strategie di controllo e realizzarle su un sistema a microprocessore utilizzando sicurezze adeguate. Per esempio il microprocessore realizzerà una nuova strategia e misurerà la rete risultante o le caratteristiche dell'incrocio. Se queste caratteristiche risultano migliori rispetto a quelle ottenute in condizioni simili durante lo stesso giorno o durante i giorni precedenti la nuova strategia verrà omologata. Se invece qualsiasi parametro risultante mostra una variazione di ampiezza anomala, l'uso di questa nuova strategia viene automaticamente interrotto, commutando su una strategia «collaudata e vera». Il vantaggio sostanziale di questo metodo è di consentire il collaudo di strategie in tempo reale mentre la sicurezza della rete è garantita da caratteristiche di ragionevolezza software. In questo modo risulta possibile provare nell'ambito di un giorno, delle tecniche che diversamente non sarebbe possibile provare oppure che avrebbero richiesto un tempo molto lungo e molto costose come hardware.

I sistemi di controllo del traffico basati su microprocessore sono attualmente divenuti la regola piuttosto che l'eccezione per le nuove installazioni nell'ambito della maggior parte degli Stati Uniti e di alcune città d'Europa. Le tecniche utilizzate per il controllo del flusso del traffico dei veicoli in un ambiente urbano sono molto simili alle tecniche per il controllo di qualsiasi flusso continuo o discreto. In California ed in qualche altro luogo le versioni modificate di questi controllori del traffico sono utilizzate per applicazioni di misura di flusso dell'acqua e di altri fluidi.

Questo esempio illustra, in termini essenziali, i vantaggi e le caratteristiche dell'impiego di un microprocessore in un ambiente di controllo industriale. Un altro caso di studio, che considereremo ora, richiede un approccio radicalmente diverso, cioè l'impiego di un chip progettato su misura.

Sistema di accensione a scintilla con controllore a microprocessore

La Delco Remy, una divisione della General Motors, ha introdotto nello scorso '76 il primo sistema di temporizzazione di scintille per automobili controllato da microprocessore. Il sistema è utilizzato sulla Oldsmobile. Un diagramma del sistema viene riportato in Figura 6-7. Il sensore fornisce al microprocessore le informazioni richieste. Gli ingressi del microprocessore sono costituiti dal vuoto del motore, dalla posizione dell'albero a gomiti, dalla temporizzazione di riferimento e dalla temperatura del liquido di raffreddamento.

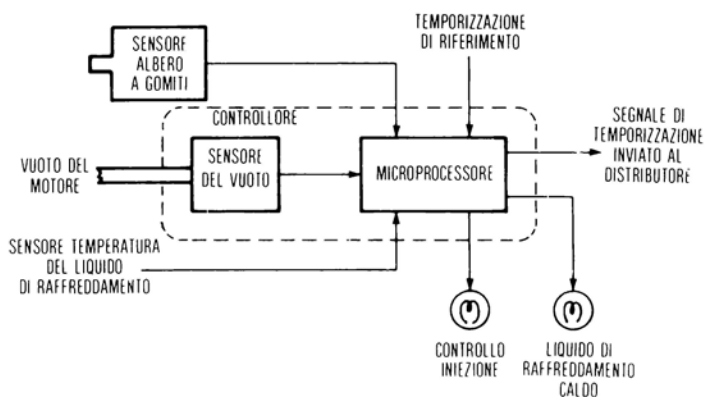



Fig. 6-7: Sistema di temporizzazione di scintille Delco Remy.

Le uscite del sistema sono tre. Quella principale è il segnale di temporizzazione inviato al distributore che sarà inviato alle candele. Le altre due uscite del sistema sono le informazioni di stato: «controllo accensione» e «caldo». Un'immagine del modulo di controllo effettivo è riportata in Figura 6-8. Il microprocessore progettato su misura è il DIP che appare al centro dell'illustrazione con un coperchio circolare.

Un microprocessore di tipo speciale è stato progettato per quest'applicazione dalla divisione automobili della Rockwell. Il sistema funziona in un modo guidato da tabella. Non esiste un semplice algoritmo che determini la temporizzazione corretta per le scintille in funzione delle condizioni d'ingresso. La generazione di questa temporizzazione è considerata un'arte. Ogni motore, una volta entrato in produzione, viene controllato in modo estensivo. I costruttori stabiliscono tabelle che determinano la temporizzazione desiderata in funzione di un insieme di parametri esterni. Il sistema MISAR (nome del sistema assegnato dalla Delco-Remy) realizza una versione automatizzata di questo meccanismo di consultazione di tabella. Un sottinsieme di tabelle viene memorizzato in una memoria speciale. Per ogni insieme di condizioni esterne misurate dal sistema vengono reperiti i relativi ingressi alla tabella. Per il calcolo dei valori intermedi viene utilizzata una tecnica di interpolazione. Speciali istruzioni realizzate all'interno di questo microprocessore costruito su misura (o custom) consentono proprio questo. Questo microprocessore è dotato di linee d'ingresso-uscita dirette come pure di caratteristiche analogiche dirette. Esso realizza le speciali funzioni di consultazione di tabella che caratterizzano l'impiego delle tabelle ROM.

La General Motors dichiara un certo numero di vantaggi per questo sistema. Senza dubbio il più significativo è la riduzione di inquinamento. Il controllo preciso della temporizzazione si traduce in un notevole miglioramento della combustione

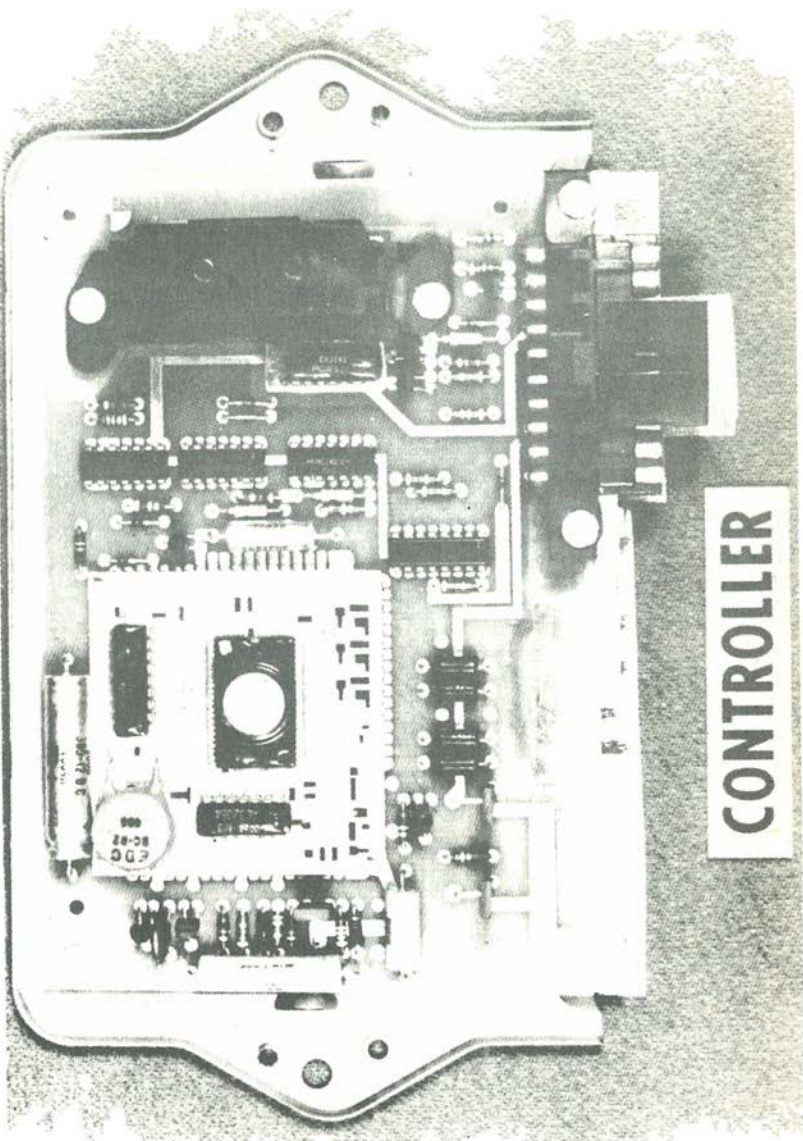



Fig. 6-8: Controllore (MISAR).

del motore e livelli di emissione di inquinanti molto più bassi. Conseguentemente ci si aspetta che l'impiego di questo sistema di temporizzazione programmata di scintilla condurrà all'eliminazione del convertitore catalitico di cui deve essere equipaggiata la maggior parte delle automobili americane. Infatti il convertitore catalitico era stato reso necessario dalle forti emissioni del caso convenzionale. Esso è un articolo costoso e la sua eliminazione condurrebbe ad un sostanziale vantaggio economico per la compagnia. Gli altri vantaggi dichiarati sono alcuni piccoli miglioramenti nel consumo di benzina e nella rispondibilità del motore. Un vantaggio significativo di questa tecnica (per il costruttore) è che durante il collaudo di massa di questo approccio programmato, il programma sarà migliorato e condurrà ad un miglioramento significativo della temporizzazione del motore nella maggior parte delle condizioni. Questo potrebbe risolversi in un miglioramento di risparmio di benzina come pure in un'ulteriore riduzione dell'inquinamento. Dopo la dichiarazione della General Motors, la maggior parte degli altri costruttori ha annunciato dei progetti per dispositivi simili. Sembra che la Ford stia lavorando in Giappone su un nuovo progetto con la Toshiba, come pure la Motorola e la Essex negli Stati Uniti e molti costruttori di microprocessori hanno annunciato il loro possibile contributo ai programmi delle automobili. Una microfotografia del microprocessore custom appare in Figura 6-9.

I microprocessori applicati nell'automobile potrebbero essere utilizzati per un certo numero di funzioni addizionali. Essi potrebbero essere utilizzati per osservare le condizioni del motore e per mostrarle al pilota. Essi potrebbero anche fornire i diagnostici. Essi possono fornire una direzione completa del pannello display dall'orologio al tachimetro digitali ed all'osservazione delle condizioni anomale del motore. L'affidabilità e le caratteristiche dei microprocessori nelle più difficili condizioni ambientali sono sufficienti per quest'applicazione. Il problema essenziale non è tanto il chip del microprocessore, ma i sensori richiesti ed i displays. Il costo dei sensori è ancora alto e la loro affidabilità non è sufficiente in condizioni ambientali difficili. Ne risulta che i costruttori di automobili si astengono dall'impiego del microprocessore poichè ogni costo, seppur piccolo, è significativo nel costo dell'automobile. Comunque è soltanto una questione di tempo finchè non verranno sviluppati i sensori ed i displays richiesti. Ci si può aspettare che le automobili, come qualsiasi altro dispositivo meccanico complesso, saranno equipaggiate di diversi microprocessori realizzanti un certo numero di funzioni, di sicurezza e funzionali, nel prossimo futuro. Analogamente alle altre applicazioni, una volta che i microprocessori sono installati nel veicolo la potenza disponibile durante gli intervalli di servizio renderà possibile un certo numero di caratteristiche di intelligenza e di convenienza che altrimenti non potrebbero mai essere ottenute. Con tutta probabilità mancano ancora diversi anni al loro impiego su larga scala.

Un controllore di forno a microonde

Un controllore di forno a microonde è caratterizzato da un algoritmo di controllo molto semplice, che tradizionalmente è stato realizzato in forma elettromeccani-

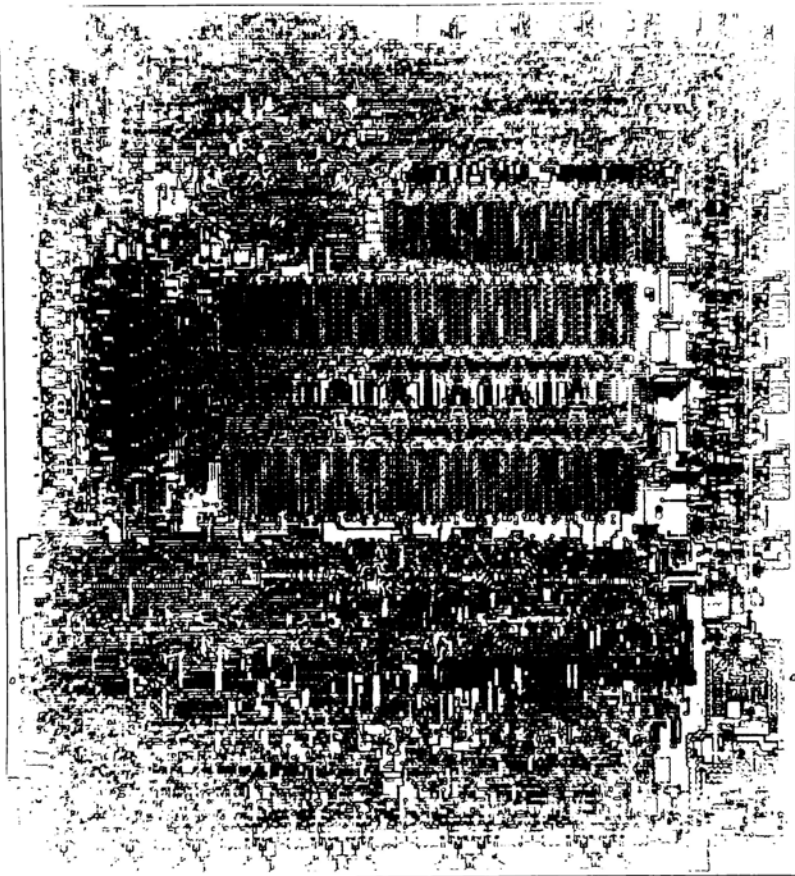



Fig. 6-9: Il microprocessore custom della Rockwell.

ca. L'elevato volume di produzione implica il più basso costo possibile per la logica di controllo. Infatti il controllo è realizzato con un microcalcolatore su chip singolo (come un F8, un 8048, un PPS4/1 oppure un TMS1000). D'altra parte, contrariamente a quanto si potrebbe pensare, uno dei costi principali nell'impiego di un tale microprocessore su chips singolo può non essere il microcomputer stesso od il display e la tastiera richiesti. In questo caso il sistema è dotato di una tastiera esadecimale. In alcuni casi il numero di tasti è esteso a 24 o più. La tastiera è un dispositivo di ingresso convenzionale. L'utente specificherà per mezzo della tastiera l'istante in cui il forno deve iniziare la cottura. Esso specificherà la natura della carne o della verdura da cuocere (per esempio «pollo») e quindi specificherà il peso (per esempio: «10 kg»). A questo punto viene attivato un allarme acustico ed una spia inizia a lampeggiare: i dati vengono scartati.

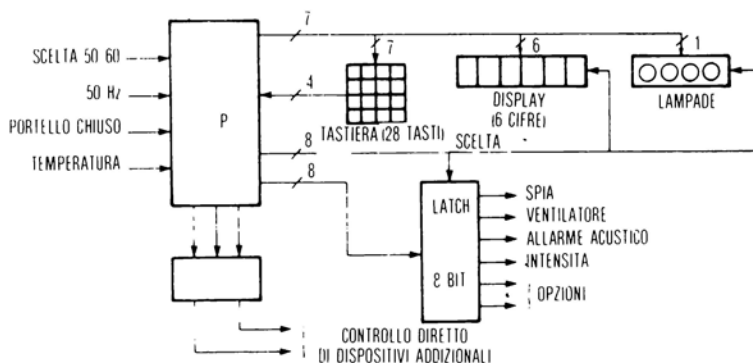



Fig. 6-10: Controllore di forno a microonde.

Questo è un controllore «intelligente» del forno. Esso impiega la prova di ragionevolezza. Esso rigetterà qualsiasi dato ritenuto irragionevole. Per un controllore di forno equipaggiato di microprocessore non esistono polli di 10 kg. L'utente viene quindi avvertito dell'errore e può introdurre nuove istruzioni. Si supponga che i dati introdotti siano ragionevoli, il controllore calcolerà automaticamente il tempo di cottura richiesto e lo realizzerà. Naturalmente questo conduce ad una grande convenienza a livello di utente e ad un costo molto piccolo a livello di costruttore. Inoltre, dato che è disponibile un microprocessore, si possono ottenere diverse altre funzioni durante l'intervallo di tempo di non impiego del display. In quantità di decine di centinaia o più il costo di un tale microcomputer a chip singolo è circa 2 dollari. Un altro vantaggio derivante dalla presenza del microprocessore è il seguente: se in una data posteriore si dovessero rendere necessarie ulteriori funzioni di controllo, esse potrebbero essere realizzate semplicemente modificando il programma sul microprocessore. Le funzioni, di automatizzazione come il tempo di cottura richiesto, sono realizzate mediante il semplice funzionamento mediante consultazione di tabella. Questo è tipico delle caratteristiche di convenienza introdotte nelle apparecchiature equipaggiate, di microprocessore. Esse automatizzano la maggior parte delle procedure tediose precedentemente richieste. Il loro valore di convenienza è molto elevato e ciò in relazione alla domanda del mercato. In questo caso occorre notare che il microprocessore non introduce una capacità di controllo addizionale rispetto alla realizzazione elettromeccanica. Il suo valore è aggiunto in convenienza. Poichè i forni a microonde sono orientati alla convenienza, questo è un punto significativo per la vendita.

Controllore di copiatrice

In Figura 6-11, come ulteriore esempio, appare la struttura di un controllore di copiatrice. È ovvia la rassomiglianza al sistema precedente. La ragione potrebbe

non essere altrettanto ovvia. Si tratta di un microcomputer a chip singolo con un ingresso a tastiera, un'uscita LED e relè. L'unica differenza significativa risiede nel ruolo delle varie linee d'ingresso e d'uscita del sistema. Comunque, nel caso di controllore di copiatrice, le funzioni di controllo addizionali, che non potevano essere considerate nel caso precedente, divengono possibili. Qualora si verifichi un malfunzionamento meccanico, quale un inceppo della carta, si possono realizzare diversi algoritmi di controllo che risolveranno i problemi di bloccaggio meccanico (come l'emissione di un foglio di carta mediante rotazione del motore in direzione inversa). Ne risulta: caratteristica di prodotto migliorata (copie migliori ed in numero maggiore), convenienza migliorata (minori bloccaggi), diagnostici migliorati e miglioramento dell'affidabilità globale. L'affidabilità globale può anche essere migliorata attraverso le routines di testing. Il microprocessore per eseguire dei controlli sulle medie degli ingressi e delle uscite della copiatrice e verificare che essi abbiano priorità operativa al funzionamento di avviamento della copiatrice. Una scheda singola di hardware può inoltre essere sviluppata per una gamma di copiatrici. Solo il programma cambierà da un modello ad un altro, originando benefici simili a quelli che erano stati sottolineati nel caso del controllore del traffico.

PERSONAL COMPUTING

A questo punto potrebbe essere probabilmente ripetitivo ed improduttivo osservare ulteriori applicazioni tipiche. È stato stabilito che la struttura del sistema computer di base è costante. L'interfaccia ed il software, cioé il ruolo giocato dal sistema, sono diversi in ogni applicazione. Comunque a questo punto merita un particolare accenno la nuova area di impiego del microprocessore: è il nuovo mercato dell'hobby.

Una conferenza tenutasi ad Atlantic City nell'estate 1976 ha mostrato che numerose persone sono coinvolte nell'impiego dei microprocessori come hobby. At-

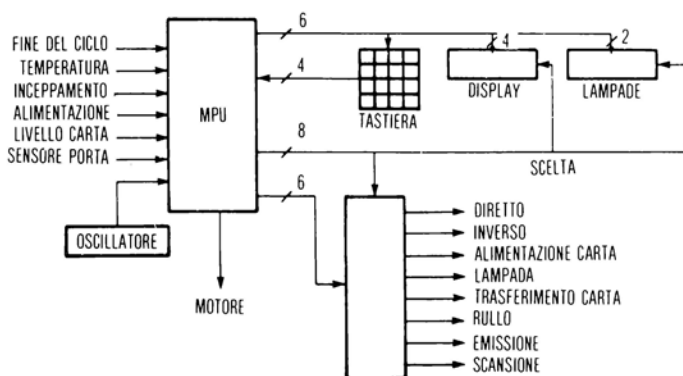



Fig. 6-11: Controllore di fotocopiatrice.

tualmente questo mercato è stato chiamato «personal computing». Con l'avvento di famiglie complete di componenti per microprocessori è stato mostrato nei precedenti capitoli come sia semplice assemblare un sistema completo. Diversi costruttori hanno realizzato questo fatto ed iniziato il mercato degli insiemi di componenti assieme ai diagrammi per facilitare l'assemblaggio. Una volta fornite le istruzioni di assemblaggio e la scheda del circuito stampato, l'assemblaggio di un computer completo funzionante, è soltanto un problema di tempo anche per un principiante. Per le persone che trovano in questo un valore educativo, ovvero di divertimento, si tratta di uno degli strumenti o giochi più sofisticati mai introdotti. Per ora studenti, ingegneri elettronici, dottori, avvocati e uomini di affari sono attivamente coinvolti nella costruzione di piccoli computer utilizzando componenti a microprocessore oppure nella loro programmazione. Si stima che, già nel 1977, essi siano già centinaia di migliaia di persone. È opinione diffusa che un ampio settore della popolazione può essere coinvolto in questo «hobby». La varietà di utenti è molto grande. Gli uomini d'affari ed i professionisti normalmente assemblano o programmano tali sistemi per un beneficio diretto quale una direzione di file, un conteggio, un aiuto alla loro professione. Altri utenti li costruiscono e li programmano semplicemente per valore di gioco e di intrattenimento su di essi. Infatti si può sospettare che la maggior parte degli utenti fa questo soprattutto per il valore gioco senza disporre di una conoscenza diretta. Si ritiene che l'impatto commerciale di questo nuovo mercato sia molto significativo. Attualmente una scheda microcomputer costa poche centinaia di dollari assemblata o meno. Il principale ostacolo all'acquisto dei sistemi è stato, come sempre, il costo delle periferiche. In vista del nuovo mercato di massa appena aperto, sono state sviluppate nuove periferiche il cui costo è di poche centinaia di dollari. Si ritiene che, nell'arco di alcuni anni, sarà disponibile un sistema completo con tutte le periferiche richieste per circa 2000 dollari o meno. Questo apre l'era dei computer in casa.

Una conseguenza tecnica di questo nuovo mercato dell'hobby è stato l'avvento di nuove convenzioni. In particolare un nuovo sistema di bus è divenuto convenzionale di fatto: è il bus S-100. Esso è il bus introdotto dalla MITS sul suo microcomputer Altair, utilizzato anche dalla ISMAI. Esso è un bus fortemente orientato all'8080 che è essenzialmente compatibile con lo Z-80 e che può essere adottato al 6800. Come conseguenza del mercato di massa sviluppatosi, attualmente la maggior parte dei costruttori di periferiche a basso costo producono dispositivi che sono direttamente compatibili col bus S-100. Tali dispositivi possono quindi essere connessi direttamente al bus. La denominazione del bus S-100 è stata scelta per indicare per la prima volta un bus standard effettivo. Esso viene esaminato in dettaglio al Capitolo 7.

Gli hobbisti per che cosa usano i microcomputer? Una risposta è la seguente: se una applicazione è già stata realizzata il vero hobbista ne realizzerà una diversa. Essenzialmente questo significa che lo spettro delle applicazioni possibili è illimitato. Alcuni semplici esempi delle prime applicazioni dei microcomputer per hobby

sono i seguenti: regolazione automatica dei sistemi innaffiatoi in casa, sistemi di allarme sofisticati, pianificazione di modelli di controllo, giochi e per il conteggio.

Un'altra conseguenza significativa degli hobbisti nel campo dei computer è stata una spettacolare caduta nel costo del software. Gli interpreter del Basic come pure i programmi più comuni sono ora disponibili ad un costo di alcuni dollari. Molti hobbisti hanno sviluppato programmi nel loro tempo libero ed attribuito grande importanza al fatto di legare il proprio nome ad un dato programma. Questo a sua volta ha causato la rovina di molte piccole software house poichè il tempo di programmazione degli hobbisti è essenzialmente libero.

Negozi di microcomputer sono ora disponibili in molte città, alimentando questo nuovo mercato. Essi inoltre smerciano ciò che ora viene chiamato «software plastico» cioè contenitori plastici di PROM che possono essere connessi a qualsiasi microprocessore standard e contengono una biblioteca di programmi utili. In questo modo il software ha acquisito la comodità di un contenitore plastico.

Nessuno può prevedere fino in fondo l'impatto di questo nuovo mercato nel campo tecnico. È certo significativo il fatto che, per la prima volta, si è aperto un mercato di tipo consumistico verso le case di semiconduttori, forzando molte di esse al commercio di sistemi. Inoltre il lavoro su applicazioni originali dei microprocessori è diretto a generare un certo numero di applicazioni sorprendenti. È certo che tutte le nuove applicazioni tenderanno a mostrare di non essere mai state realizzate in precedenza in modo da risultare sorprendenti.

SOMMARIO

Sono state presentate, all'interno di questo capitolo le quattro principali aree di applicazione per i microprocessori. Sono stati quindi descritti esempi di studio illustrando i vantaggi e gli svantaggi di applicazioni tipiche dei microprocessori. È stato dimostrato che è semplice la costruzione progressiva di un'applicazione mediante l'aggiunta di moduli. L'originalità di qualsiasi progetto risiede nell'interfacciamento e soprattutto nella programmazione dell'applicazione. Verranno ora esaminati questi due settori.

CAPITOLO 7

TECNICHE DI INTERFACCIAMENTO

SCOPO

L'interfacciamento di un sistema microprocessore a dispositivi esterni coinvolge sia le tecniche hardware che software. Durante la progettazione di un nuovo sistema esiste sempre un compromesso tra hardware e software. Generalmente i componenti hardware amplificheranno il progetto e forniranno caratteristiche migliori. Comunque ciò aumenta il numero di componenti. Le tecniche software alternative elimineranno i componenti hardware conducendo ad un costo più basso del sistema. Comunque, in generale, esse condurranno ad un aumento della complessità software e degraderanno le caratteristiche. In questo capitolo saranno presentate le tecniche essenziali ed i componenti LSI attualmente disponibili per interfacciare i sistemi microprocessori ai dispositivi esterni. La gamma di tali interfacce tipiche andrà da una semplice interfaccia di tastiera ad un'interfaccia di floppy-disk. Nel caso della maggior parte dei dispositivi complessi, non sarà possibile entrare in considerazioni tecniche dettagliate e si consiglia vivamente al lettore di fare riferimento al nostro libro sulle tecniche di Interfacciamento. Questo capitolo focalizzerà la realizzazione hardware delle tecniche descritte. L'utente dovrebbe sempre ricordare che si può sempre far ricorso alla realizzazione software di tutte le parti delle tecniche descritte.

Ora verranno interfacciati ad un sistema microprocessore standard dei dispositivi progressivamente più complessi: una tastiera, un display LED, una telescrivente, un floppy disk ed infine un display CRT. Successivamente verranno presentate alcune considerazioni sull'interfacciamento di multimicroprocessori. Infine saranno descritti gli standards dei bus: IEEE 488, IEEE 583 (CAMAC) e del bus S-100 che è stato introdotto alla fine del capitolo precedente.

TASTIERA

Occorre distinguere due categorie di tastiere: le tastiere non codificate e quelle interamente codificate.

Le **tastiere interamente codificate** forniscono automaticamente il codice ASCII corrispondente al tasto che è stato premuto. Quindi le parti elettroniche relative devono *rivelare* automaticamente il tasto (normalmente 64 o più tasti) che, una

volta *identificata, fornisce il codice* corrispondente. Inoltre viene normalmente fornito uno strobe. Infine saranno descritte le normali caratteristiche di cui queste tastiere devono essere dotate, cioé la protezione contro il *rimbalzo* ed il *roll-over di tasti multipli.*

Una tastiera interamente codificata è naturalmente la più facile da impiegare in un sistema in quanto essa esegue tutte le operazioni che diversamente devono essere realizzate dal software o da componenti addizionali. Comunque, a causa del costo delle parti elettroniche associate, una tastiera interamente codificata è dispendiosa. Conseguentemente il suo impiego è ristretto a due fondamentali aree di applicazione:

1. Ogni volta che viene considerato un piccolo numero d'unità, cioé quando il costo hardware é dominante, dovrebbero essere utilizzati i sistemi interamente codificati.

2. Le tastiere interamente codificate sono utilizzate quando la tastiera stessa è complessa, cioé comprende 64 o più tasti (tastiera interamente alfanumerica).

D'altra parte la tastiera **non codificata** è probabilmente il mezzo d'ingresso più economico attualmente disponibile per l'ingresso ai microprocessori. Essa fornisce semplicemente una matrice di righe e di colonne. Ogni cosa viene realizzata mediante programma. Si vedrà che attualmente sono disponibili nuove alternative hardware.

Verrà ora descritto l'interfacciamento diretto ad una tastiera non codificata. Tutte le tecniche descritte possono essere realizzate mediante software. Si vedrà che alcune di queste, o tutte, possono anche essere realizzate utilizzando nuovi chips di interfaccia LSI. Devono essere risolti quattro problemi:

1. Identificazione del tasto
2. Generazione del codice corrispondente
3. Rimbalzo
4. Protezione contro il roll - over.

Verranno ora esaminati questi problemi e le loro soluzioni:

Identificazione del tasto

Per identificare il tasto che è stato premuto sono impiegate due tecniche di base. Il metodo tradizionale è la *scansione di riga* ed il metodo più recente, reso possibile da componenti come il PIO, è la tecnica di *linea inversa*.

Metodo 1: scansione di riga

In Figura 7-1 sono riportate le quattro fasi impiegate nella scansione di riga di una tastiera 4x4. Il tasto nero è quello che è stato premuto. Il problema consiste nella rivelazione del tasto che è stato premuto. Questo problema può sembrare semplice. In realtà non è così semplice come sembra.

In questo primo esempio non saranno rispettate le polarità effettive. Verranno riservate le polarità d'ingresso reali. Questo dettaglio, qui verrà trascurato per scopi di chiarezza. In una tastiera non codificata ogni tasto non è connesso ad un unico conduttore. Se ogni tasto fosse connesso ad un conduttore dovrebbe essere semplice identificarlo. Per produrre tastiere a basso costo, vengono forniti soltanto otto conduttori, quattro conduttori per le righe e quattro per le colonne. La pressione di un tasto connette semplicemente una riga ad una colonna. Qui si utilizzeranno le colonne come uscite e le righe come ingressi ad una porta microprocessore. Si supponga di avere sulla porta del microprocessore il valore d'uscita «1 1 1 1». Ogni colonna assume il valore «1».

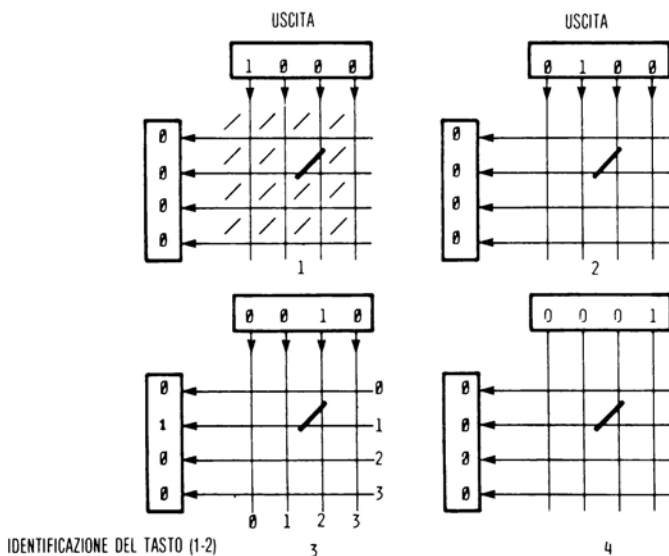



Fig. 7-1: La «scansione di riga» viene impiegata per identificare il tasto.

Il valore risultante letto sull'ingresso sarà: «0 1 0 0». Si individuerà cioè un «1» sulla riga 1 a causa della sua connessione alla colonna 2, realizzata dalla chiusura del tasto. Sfortunatamente questa informazione non ci dice quale *colonna* è stata attivata. Non esiste un modo semplice che, per mezzo di un solo insieme di dati, determini *quale colonna* e *quale riga* sono connesse. Per determinare qual'è la colonna del tasto premuto, sarà necessario fare uscire un «1» successivamente in tutte le colonne: colonna 0, colonna 1, colonna 2, colonna 3. Si eseguirà cioè la *scansione delle colonne*. Il nome di questa tecnica è «scansione di riga». In questo esempio sono state utilizzate le colonne al posto delle righe.

Si consideri il seguente esempio. Si faccia riferimento alla Figura 7-1. Scrivendo «1 0 0 0» nell'appropriata porta d'uscita, si farà uscire un «1» sulla colonna 0. L'ingresso risultante sarà «0 0 0 0» come indicato sul diagramma 1 di Figura 7-1.
L'uscita successivamente generata sarà «0 1 0 0», in corrispondenza di un «1» scritto sulla colonna 2. L'ingresso risultante al microprocessore è ancora «0 0 0 0».

La terza fase è rappresentata nello schema 3 della Fig. 7-1: l'uscita è ora «0 1 1 0». In questo modo sono state identificate la colonna 2 e la riga 1. Il tasto è identificato.

Comunque è necessario continuare la scansione a causa del problema del *rollover*.

Rollover significa che un utente può aver premuto accidentalmente e contemporaneamente più di un tasto. Questo fatto deve essere rivelato. La scansione deve perciò procedere alla sua quarta fase dove viene fatto uscire un «1» sulla colonna 3. Poichè in questo esempio non si è verificato il rollover, il valore d'ingresso risultante è «0 0 0 0» ed il processo di «scansione di riga» è terminato: non è stata rivelata nessun'altra chiusura di tasto.

Il problema successivo è la generazione del codice corrispondente al tasto, cioè la conversione di «0010-0100» in un opportuno codice a 4 bit. Questo sarà semplicemente realizzato da una consultazione di tabella su ROM. I bit di identificazione del tasto «0010-0100» saranno utilizzati per indirizzare i contenuti di una tabella dei «codici dei tasti» allocata in memoria.

Ogni volta che esiste una condizione di rollover, cioè quando viene rivelata più di una chiusura di tasto, la soluzione più semplice è di ignorare i dati e di eseguire la lettura solo se viene rivelato un segnale singolo, indicante che è stato premuto un solo tasto. Ogni volta che più di due tasti sono premuti contemporaneamente, sorgono dei problemi addizionali che richiedono l'impiego di diodi di interconnessione (vedere Illustrazione 7-2).

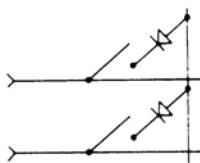



Fig. 7-2: Nel caso di roll-over complesso si devono aggiungere dei diodi.

Metodo 2: linea inversa

L'interfaccia parallela universale, il PIO, ha reso possibile l'impiego di una nuova tecnica per l'identificazione del tasto: la linea inversa è il metodo più veloce e più

elegante della scansione di riga ed è possibile ogni volta che una porta completa ad 8 bit di un PIO può essere dedicata all'interfacciamento della tastiera.

È essenziale ricordare che ogni linea di una porta di un PIO, od ogni gruppo di linee, è programmabile indipendentemente come un ingresso o come un'uscita. La Fig. 7-3 illustra l'impiego di un PIO per l'identificazione del tasto.

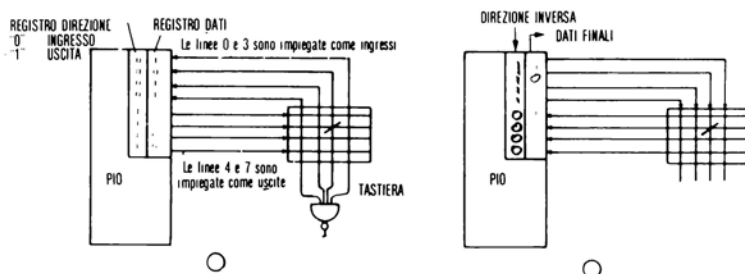



Fig. 7-3: La «inversione di linea» richiede un PIO.

L'identificazione viene eseguita in due fasi

Nella prima fase, le connessioni delle quattro righe del PIO sono programmate come uscite. Le connessioni delle quattro colonne sono programmate come ingressi. Il valore «0 0 0 0» è quindi fatto uscire sulla porta del PIO. Il valore risultante sull'ingresso è: «1 0 1 1» (questa volta sono utilizzate le polarità effettive). Lo zero corrisponde alla posizione di riga dove il tasto è stato premuto, connettendo a massa la colonna corrispondente.

Nella seconda fase vengono invertiti gli ingressi e le uscite. Questo scambio degli ingressi e delle uscite può essere realizzato molto semplicemente con un PIO. Si tratta di cambiare i bit del Registro di Direzione Dati da 0 ad 1 e da 1 a 0. Il contenuto iniziale del registro di direzione dati era «0 0 0 0 1 1 1 1». Esso viene cambiato in «1 1 1 1 0 0 0 0». Questo può essere eseguito da una semplice istruzione «Complementa il Registro di Direzione Dati». Nessun cambiamento viene eseguito sul registro dati stesso.

Vengono quindi letti i contenuti risultanti del registro dati: dalla Fig. 7-3 si può vedere che l'ingresso è ora «1 0 1 1». Lo «0» corrisponde alla riga dove è stato premuto il tasto. Il microprocessore legge i contenuti completi del registro dati che sono: «1 0 1 1 1 0 1 1», dove ogni «0» denota una chiusura di tasto. In questo modo viene identificata la riga e la colonna. Il codice ad 8 bit che è stato letto può essere utilizzato come vettore di diramazione ad una tabella ROM che contiene il codice ad 8 bit corrispondente alla tastiera. Qualsiasi salto ad una locazione non consentita potrebbe essere rivelato come un roll-over oppure un problema di rumore e si do-

vrebbe verificare una nuova lettura. Fondamentalmente questo metodo richiede solo quattro istruzioni. Esso è molto più efficiente di quello precedente.

Il principale svantaggio di questa tecnica è di dover dedicare 8 pins di un PIO alla direzione di una tastiera. A causa dell'elevato costo di un PIO rispetto al costo di una tastiera, tale soluzione può non essere accettabile e potrebbe essere preferito l'uso di decodificatori. Comunque spesso sono disponibili 8 pins ed allora questa soluzione e realizzabile.

Il problema dell'eliminazione del rimbalzo

In tutti i componenti elettromeccanici che coinvolgono un contatto, la vera chiusura del contatto si verifica solo dopo un periodo di oscillazione di diversi millisecondi. Normalmente trascorrono da 10 a 20 millisecondi tra l'istante in cui viene premuto il tasto e quello in cui viene realizzato il contatto. Lo stesso problema si verifica nel momento in cui il tasto viene rilasciato. La Figura 7-4 illustra i rimbalzi dei gradini di inizio e fine. Una semplice soluzione hardware al problema è l'impiego di un filtro RC. Ogni volta che si ha un piccolo numero di tasti questa è la soluzione tipica.

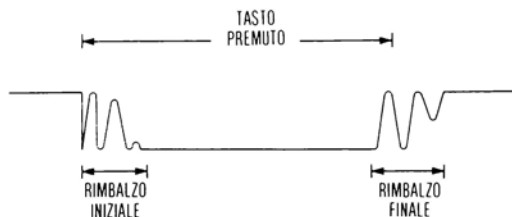



Fig. 7-4: Il problema del rimbalzo.

La soluzione software è di controllare la chiusura del tasto ancora dopo n msec che questo è stato rilasciato, dove n = da 5 a 20 (msec) in dipendenza della qualità della tastiera utilizzata. La tecnica di eliminazione di rimbalzo per via software è spesso utilizzata quando il numero di tasti sale a 16 o più. Essa viene realizzata semplicemente utilizzando una routine di ritardo software. I ritardi saranno descritti al Capitolo 8.

Tastiere complesse

Qualsiasi tastiera complessa comprendente un grande numero di tasti, per esempio 64 come in Fig. 7-5, può essere decodificata mediante, uno dei metodi precedenti. Per realizzare la connessione ad una porta a 4 bit del microprocessore diviene necessario utilizzare sull'ingresso un decodificatore da 4 a 16 bit. Questo è illustrato in Figura 7-5.

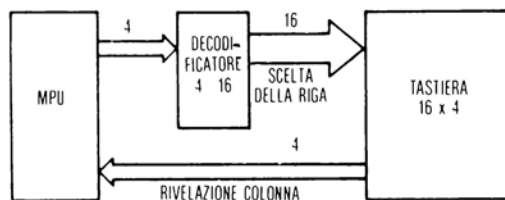



Fig. 7-5: Aggiunta di più tasti.

Qualora vengano impiegate tastiere alfanumeriche complete comprendenti un numero elevato di tasti è desiderabile impiegare anche i tasti «scorrimento» e «controllo». Questo origina quattro modi di funzionamento ovvero 4 × 64 caratteri possibili. Quindi la gestione della tastiera diventa complessa e comprende routines software lunghe e complesse ed anche decodificatori hardware addizionali. A questo punto è vantaggioso considerare i nuovi componenti speciali LSI sviluppati specificamente per tastiere e controllo di display.

Per esempio il nuovo controllore di tastiera/display 8279, prodotto dalla Intel, interfaccia direttamente una tastiera a 64 tasti ed un display LED con 16 LED doppi. Il sistema è illustrato in Figura 7-6. La figura 7-7 riporta invece l'architettura interna del dispositivo. Esso fornisce automaticamente l'eliminazione del rimbalzo della tastiera e realizza la scansione di riga. Esso comprende internamente una

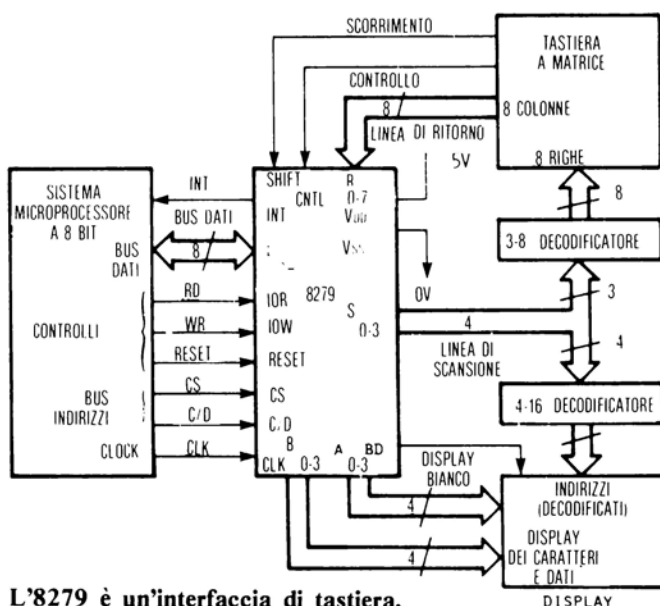



Fig. 7-6: L'8279 è un'interfaccia di tastiera.

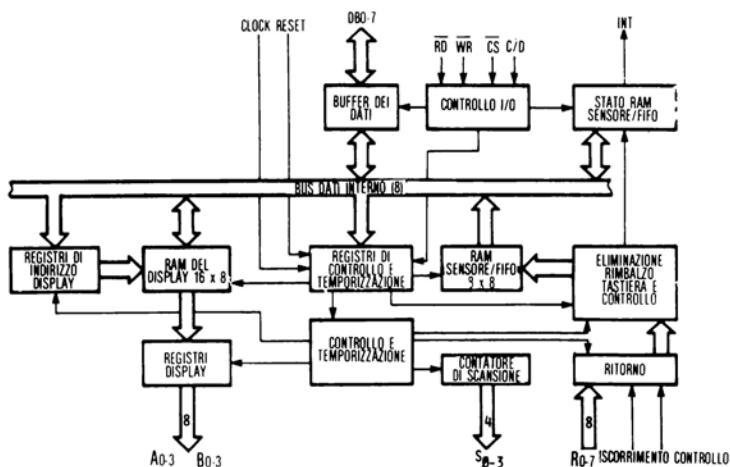



Fig. 7-7: Struttura 8279.

RAM del display 16 × 8 ed un FIFO 8 × 8. Un FIFO è una struttura First-In-First Out (primo-ingresso-prima-uscita) cioè in pratica una lista di attesa di bytes che sono accumulati dalla tastiera. Esso non fornisce le funzioni di codifica dalla tastiera, cioè la sostituzione di un codice ad 8 bit per ogni tasto premuto.

Un *codificatore di tastiera*, come il NEC-UPD 364DO2 fornisce la scansione, l'eliminazione di rimbalzo, e la decodifica della tastiera. Esso fornisce un codice a 10 bit per un massimo di 90 tasti, in uno qualsiasi di quattro modi, impiegando una ROM interna di 3600 bit. La Figura 7-8 riporta l'architettura del dispositivo. Esso può funzionare in quattro modi diversi: normale, scorrimento, controllo e scorrimento più controllo. Altri codificatori di tastiera sono disponibili da vari produttori quali la Rockwell.

DISPLAY LED

LED significa Light-Emitting-Diode (diodo emettitore di luce). La Figura 7-9 riporta un LED. Esso è composto di sette segmenti dei quali viene illuminata una combinazione per mostrare una cifra specifica. Ciascun segmento è contraddistinto da una lettera da A a G (vedere Fig. 7-9). Per esempio, per illuminare uno 0, verranno selezionati i segmenti A, B, C, D, E, F.

Per illuminare 1 verranno selezionati i segmenti B, C e così via.

Per mostrare i digits esadecimali su un LED è necessario: innanzi tutto selezionare il componente LED e quindi selezionare la combinazione di segmenti che deve essere illuminata. Inoltre, nel caso più comune, il numero da mostrare su un LED è

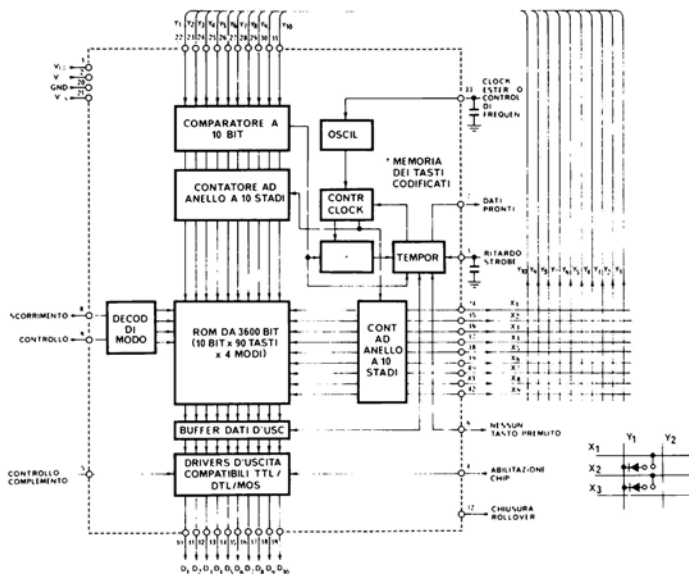



Fig. 7-8: Codificatore di tastiera NEC.

codificato nel sistema BCD. Esso deve quindi essere decodificato da BCD al codice a sette segmenti. Diversi sono i metodi che realizzano questa decodifica: mediante una memoria ROM oppure mediante un decodificatore da BCD a 7 segmenti. La Figura 7-10 riporta un esempio con un decodificatore. Ciascuna linea che alimenta un segmento deve comprendere un driver. Mediante un codice a tre bit è possibile

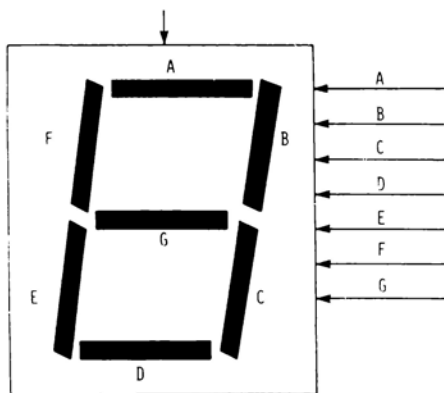



Fig. 7-9: Display LED a 7 segmenti.

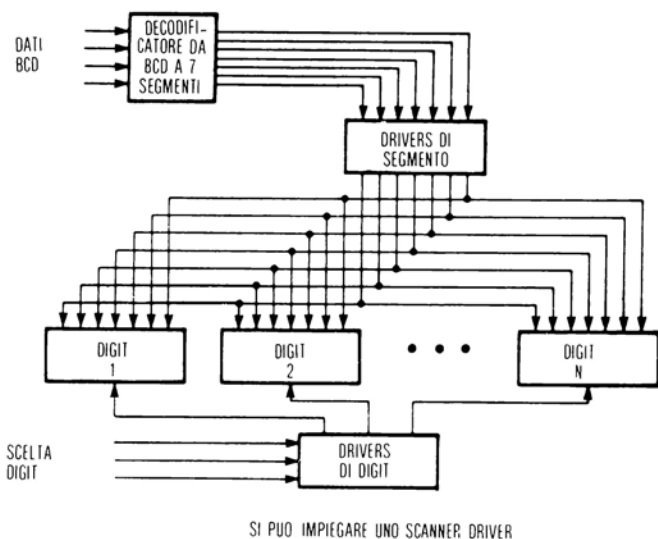



Fig. 7-10: Interfaccia LED.

selezionare un LED su otto. In particolare, nell'esempio di Fig. 7-10, viene utilizzato un driver a scansione per minimizzare il numero di componenti.

INTERFACCIA DELLA TELESCRIVENTE

Una telescrivente è un dispositivo seriale. Essa codifica ogni carattere in un codice a 7 bit più il bit di parità e quindi origina un codice di carattere di 8 bit. La trasmissione a, oppure da, una telescrivente è asincrona e sono impiegati dei bit per denotare l'inizio e la fine di un carattere. La convenzione della telescrivente standard è di impiegare 1 bit di inizio e 2 di arresto. Il formato del segnale appare in Fig. 7-11.

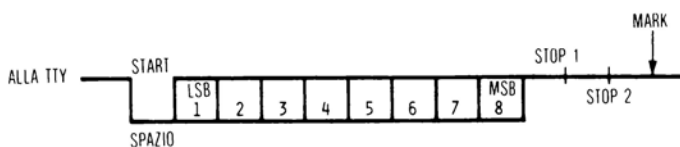



Fig. 7-11: Formato della trasmissione TTY.

Si hanno quindi 11 bit per carattere di cui solo 7 sono impiegati per l'identificazione. Una telescrivente standard trasmette 10 caratteri al secondo ed ha quindi

una velocità di 110 baud. Infatti per ogni carattere sono richiesti 11 bit ed al secondo vengono trasmessi 10 caratteri. La velocità di trasferimento massima vale perciò $10 \times 11 = 110$bps = 110 baud (nel caso di parole binarie 1 baud = 1 bps).

Per la conversione serie/parallelo e per la connessione ad un anello di corrente 20 mA della telescrivente è richiesta un'interfaccia. Alcuni modelli di telescriventi forniscono un'interfaccia diretta RS-232.

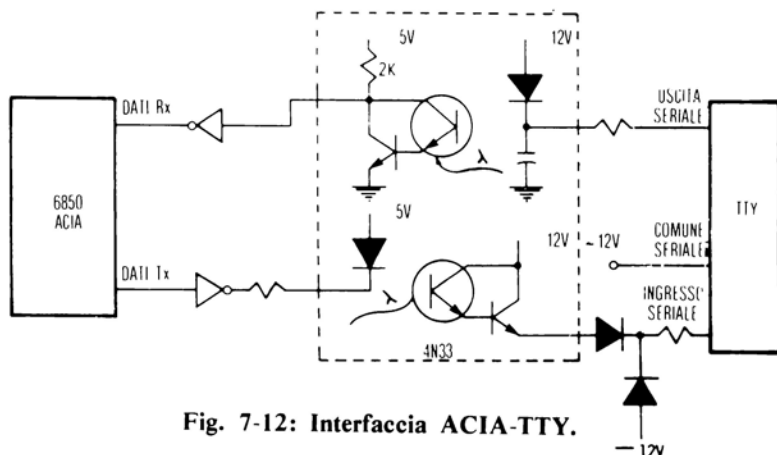



Fig. 7-12: Interfaccia ACIA-TTY.

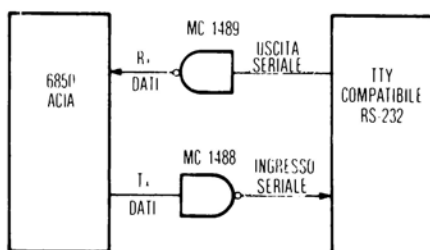



Fig. 7-13: Interfaccia ACIA-RS232.

L'interfacciamento della telescrivente coinvolge, per la conversione serie-parallelo e parallelo-serie, poco più di un UART più l'opportuna conversione di livello per l'interfaccia standard richiesta, come l'anello di corrente o RS-232. Le Figure 7-12 e 7-13 riportano queste due interfacce. Il 6850 è la versione Motorola di un UART denominato ACIA dalla Motorola stessa. La Fig. 7-14 riporta un diagramma di temporizzazione più dettagliato di una telescrivente con la sua unità lettore-perforatore di nastro.

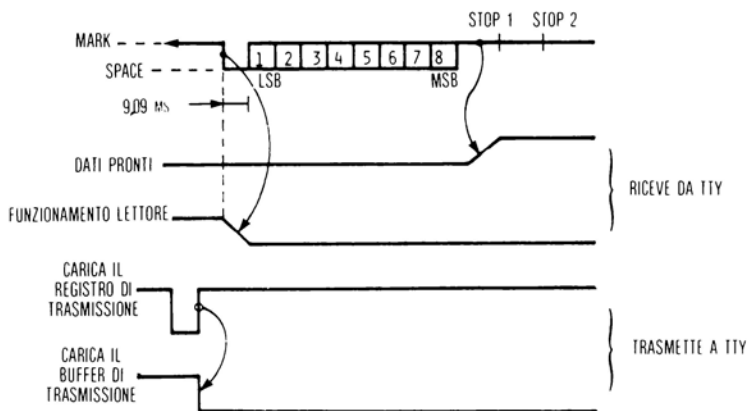



Fig. 7-14: Dettagli sulla temporizzazione.

Verrà ora descritta la sequenza di ricezione e trasmissione per la telescrivente.

Ricezione

1. Viene attivata la tastiera della telescrivente (il lettore del nastro di carta è bloccato).
2. Finchè la tastiera è inattiva fornisce un segnale di riferimento («1»); si tratta di un anello di corrente a 20 mA.
3. Quando viene premuto un tasto, viene trasmesso il codice di carattere di 8 bit più i bit d'inizio e fine; la telescrivente trasmette un bit di inizio (cioé «0») e l'ingresso dell'UART viene fatto scattare da «1» a «0».
4. L'UART registrerà ora i successivi otto bit dati e genererà un segnale di «data-ready» (dati pronti) per il microprocessore. Esso verifica la presenza di due bit di arresto e quindi li ignora. Qualora venga impiegata la parità, essa verrà

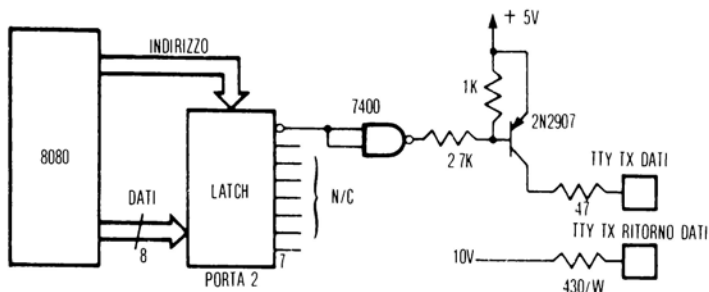



Fig. 7-15: Interfaccia 8080-TTY.

verificata dall'UART. La frequenza di clock dall'UART è 16 × 110 = 1,76 kHz.

Trasmissione

1. Il microprocessore carica il buffer d'uscita dell'UART se esso è libero.
2. Il buffer d'uscita conserva la parola dati da inviare in modo seriale alla telescrivente. L'UART genera automaticamente il formato di questa parola dati inserendo i bit di inizio e fine e la invia in modo seriale alla telescrivente.

Non è necessario impiegare un UART per la conversione da serie a parallelo. Questa funzione può essere eseguita per via software. Per esempio in Fig. 7-15 è riportata un'interfaccia impiegante un 8080 più un latch senza UART. In Fig. 7-16 è rappresentato il corrispondente diagramma di flusso funzionale.

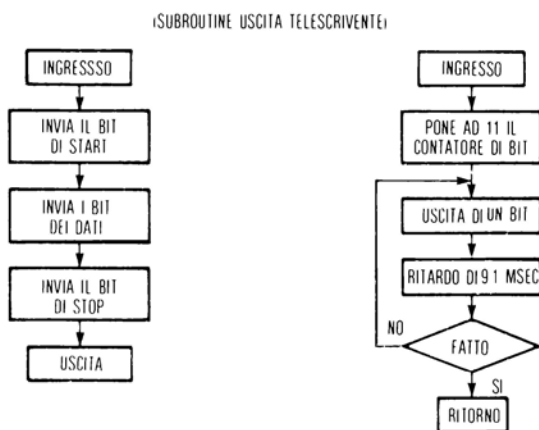



Fig. 7-16: Diagramma di flusso dell'uscita TTY.

AGGIUNTA DI UN MODEM

Un modem è un modulatore-demodulatore attualmente disponibile nel formato di chip singolo. Un modem consentirà quindi la trasmissione di dati seriali sotto forma di frequenze udibili nelle linee telefoniche. Nelle Figure 7-17 e 7-18 è riportato, a titolo di esempio, il diagramma di flusso per la ricezione e trasmissione dei dati (formato RS-232).

FLOPPY - DISK

Il floppy-disk, ovvero le più recenti versioni mini- o micro-floppy, è una delle me

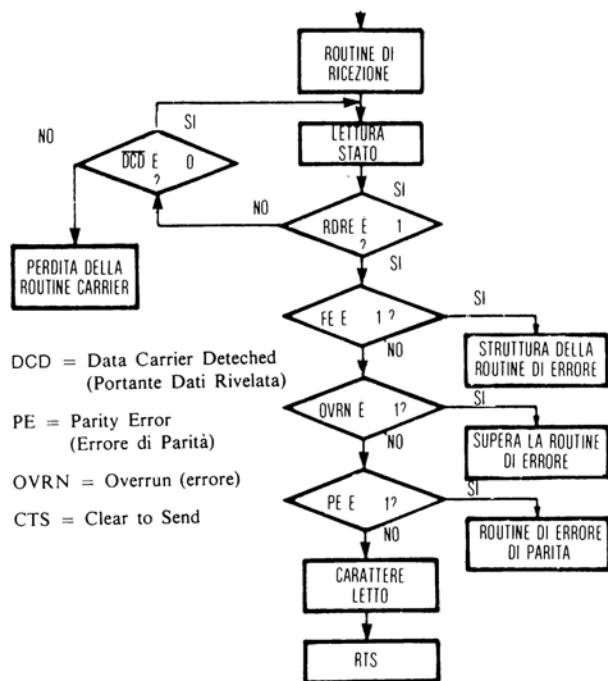



Fig. 7-17: Subroutine di ricezione.

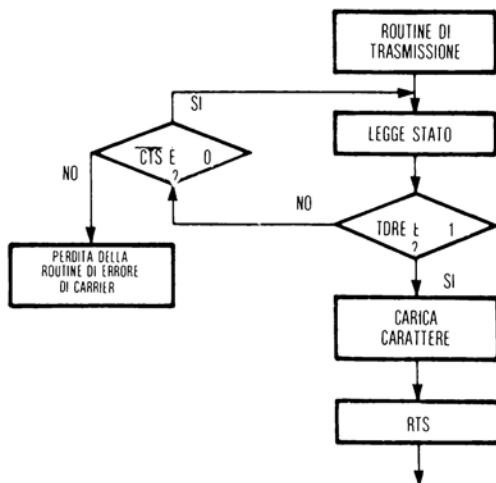



Fig. 7-18: Subroutine di trasmissione.

morie di massa più impiegate per i microprocessori. Un tipico mini-floppy fornisce 110 K bytes con una velocità di trasferimento di 125 K bps su un dischetto di 5,25 pollici (circa 13,35 cm) ed un consumo di potenza da 7,5 a 15 watt. In Figura 7-19 è rappresentato un disco tipico e la sua formattazione.

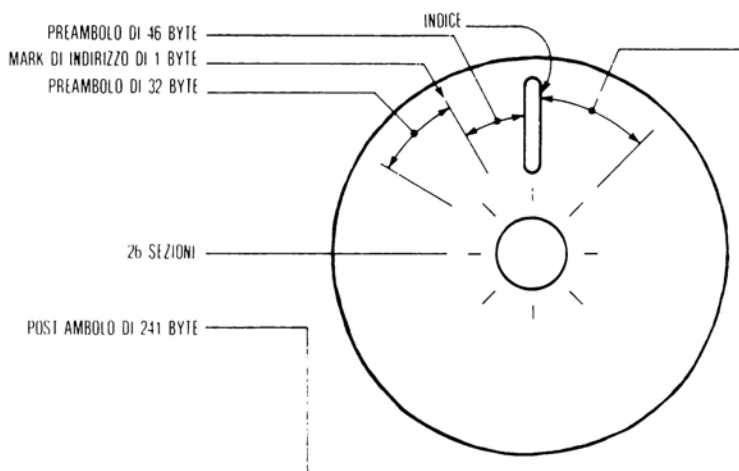



Fig. 7-19: Formato di floppy - disk.

Il nome «floppy»-disk deriva dal fatto che il disco stesso è realizzato in materiale flessibile. Esso è codificato magneticamente su uno od entrambi i lati. Esso è racchiuso permanentemente in un involucro a quadrato di cartone, foderato internamente con uno speciale materiale anti-frizione. Durante l'impiego il floppy-disk ruota, all'interno del suo involucro, a velocità elevata, in corrispondenza della testina di lettura/scrittura. La testina di lettura/scrittura si muove orizzontalmente lungo il raggio del floppy-disk in corrispondenza di un foro praticato sull'involucro detto «foro indice» (vedere Fig. 7-19).

L'informazione memorizzata nel disco è strutturata in *tracce* e *settori*. Il disco è diviso in reticoli concentrici chiamati *tracce*. Le tracce sono numerate tipicamente da 0 a 76. Le tracce sono divise in settori radiali, come spicchi di torta. Ogni blocco all'interno di un settore ha tipicamente 128 bytes di informazioni utili.

Per accedere all'informazione contenuta all'interno del disco è necessario ricercare il settore su cui esso è memorizzato e quindi trasferire l'informazione a, oppure da, esso. La testina deve essere meccanicamente posizionata sopra la traccia desiderata. Occorre quindi attendere finché il settore corretto viene a posizionarsi sotto la testina. L'informazione può essere trasferita in modo seriale. Un *controllore*

del disco ha il compito di accedere automaticamente al blocco di dati sul disco. Inoltre esso esegue diversi altri compiti come spiegato di seguito.

Una sequenza di lettura di base è la seguente:

1. Accesso alla traccia desiderata mediante posizionamento su di essa della testina.
2. Identificazione del settore iniziale (in un trasferimento a settore multiplo).
3. Trasferimento del numero richiesto di settori nella memoria del sistema.
4. Convalida CRC. CRC è il *Cyclic Redundancy Check* (Controllo ciclico di ridondanza). Esso consiste di una semplice tecnica algoritmica impiegata per verificare l'integrità dei dati con le parole precedenti memorizzate alla fine di ogni blocco logico. Occorre sia disponibile un circuito speciale per verificare il CRC, ricalcolandolo e quindi confrontandolo con uno letto dal disco.

CONTROLLORI DI FLOPPY - DISK

Dal 1977 è disponibile su chip singolo il controllore di disco completo. I floppy disks regolari sono detti a *settori flessibili* (soft-sectored) in quanto l'utente può definire il formato sul disco. Per contro i più recenti mini-floppys sono a *settori rigidi* (hard sectored). Attualmente non sono ancora disponibili i controllori su chip singolo per i mini-floppys.

Tutti gli FDC (Controllori di Floppy-Disk) devono fornire un certo numero di funzioni comuni. Essi sono tutti compatibili con l'IBM 3740. Inoltre essi forniscono sempre le funzioni seguenti:

1. Verifica automatica.
2. Compatibilità di formato.
3. Generazione e verifica CRC.
4. Lettura o scrittura con: blocchi semplici o multipli, ricerca automatica di settore, lettura o scrittura di traccia completa.
5. Inoltre essi possono fornire contemporaneamente il controllo di diversi dischi.

La conservazione della traccia di dove risiede l'informazione sul disco è in ogni caso sotto la responsabilità del programmatore. I Controlli Programmabili sono: tempo di passaggio da traccia a traccia, tempo di posizionamento della testina, tempo di aggancio della testina, controllo del motore passo-passo od a 3 fasi, trasferimento DMA programmato. È indispensabile l'impiego di un DMA per tutti i dischi di velocità standard per l'aggancio software del microprocessore con la velocità di trasferimento richiesta.

Un FDC su chip singolo

A titolo di esempio verrà descritto di seguito il controllore di floppy-disk su chip singolo della Western Digital. Si tratta del FD 1771B.

In Fig. 7-20 è riportata l'architettura interna del componente.

I quattro circuiti fondamentali sono:

1. La logica CRC (che appare in basso a sinistra nella figura). Essa genera automaticamente il CRC per l'informazione che entra o che esce. Durante il funzionamento di lettura, il CRC calcolato verrà confrontato al CRC letto dal disco. Se essi sono in accordo i dati saranno ritenuti validi.

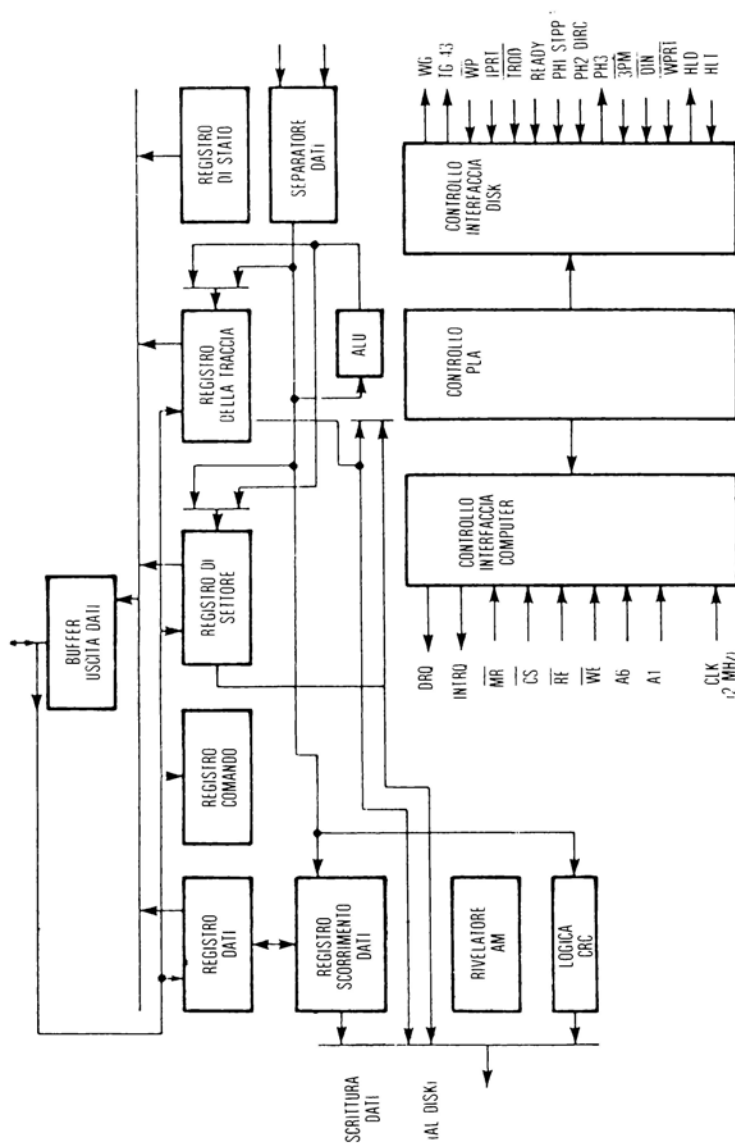



Fig. 7-20: FD 1771B della Western Digital.

2. La ALU, che appare sulla destra dell'illustrazione, è impiegata per confrontare, incrementare o decrementare registri. I registri verranno descritti di seguito.
3. Il controllo - interfaccia - disco che dirige le linee di controllo richieste.
4. L'interfaccia-microprocessore che realizza il dialogo handshake richiesto con il sistema.

Le due interfacce precedenti appaiono in Figura 7-20. Di seguito esse verranno descritte in dettaglio.

I registri FDC

I sei registri interni FDC appaiono alla sommità della Figura 7-20.

1. Il registro scorrimento-dati accumula gli otto bit dei dati dal floppy-disk e li invia al registro dati. Inversamente esso preleverà 8 bit dal registro dati e li disporrà in ordine seriale per scriverli sul disk.
2. Il registro dati è semplicemente un registro buffer bidirezionale ad 8 bit impiegato per le funzioni di lettura e scrittura sul bus dati del microprocessore.
3. Il registro traccia memorizza la posizione effettiva della testina (da 0 a 76). Esso viene incrementato durante il movimento verso la traccia 76 e decrementato durante il movimento verso la traccia 0.
4. Il registro - settore contiene il numero del settore richiesto.
5. Il registro-comando contiene il codice di comando ad 8 bit. Tale codice è stato caricato dal programmatore attraverso il bus dati del sistema. Esso contiene le opzioni relative alle possibilità di controllo dell'FDC.

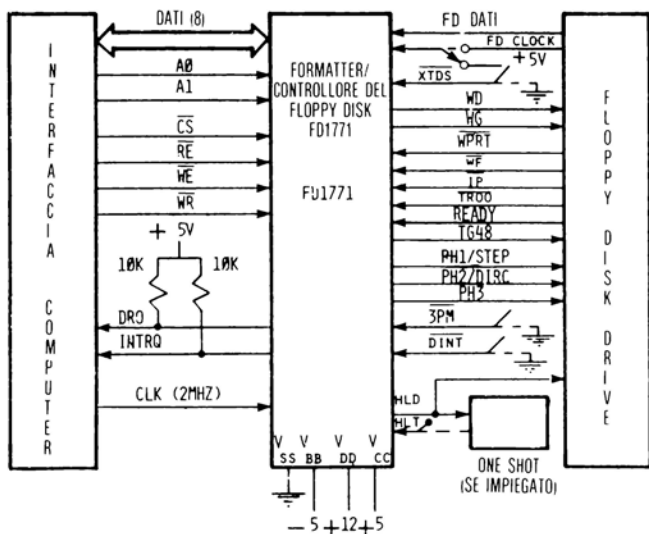



Fig. 7-21: Interfaccia FDC.

6. Il registro-stato contiene i segnali di stato generati dal componente. Esso può essere letto sul bus del microprocessore.

Interfaccia del processore

L'interfaccia del processore è riportata sulla sinistra della Fig. 7-20 e più dettagliatamente in Fig. 7-21. Essa connette il bus dati bidirezionale del microprocessore riportato alla sommità dell'illustrazione. Si ha un segnale d'uscita ogni volta che sono abilitati CS (Chip Select) ed RE (Register Enable). La Tabella 7-22 riporta i segnali. Inversamente si ha un ingresso ogni volta che sono attivi CS e WE (Write Enable).

Le linee A0 ed A1 specificano la destinazione. CS seleziona il componente FD 1771. RE e WE specificano un accesso di lettura o scrittura e sono impiegati per la selezione del registro come indicato in Tabella 7-22.

A1	A0	$\overline{RE}$	$\overline{WE}$
0	0	REG. DI STATO	REG. DI COMANDO
0	1	REG. DELLA TRACCIA	REG. DELLA TRACCIA
1	0	REG. DI SETTORE	REG. DI SETTORE
1	1	REG. DATI	REG. DATI

• L'USCITA RICHIESTA DATI (DRO) È IMPIEGATA PER DMA

Fig. 7-22: Scelta di funzione.

Il segnale DRO indica una richiesta dati in uscita (data-request-output) ed è impiegato dal DMA. Il segnale INTRQ indica una richiesta di interrupt (interrupt request) ed è attivato da diverse combinazioni di condizioni all'interno della FDC.

Interfaccia del floppy-disk

Sulla parte destra della Fig. 7-20 sono riportate le linee di interfaccia del floppy-disk. Sono disponibili le seguenti caratteristiche:

1. Controllo di posizionamento della testina.
2. Controllo di scrittura.
3. Trasferimento dati.

Viene impiegato un clock ad onda quadra di 2 MHz (internamente diviso per 4).

La testina mobile può essere programmata in tre incrementi di passo: 100, 125 e 166 passi al secondo. Il numero di passi al secondo è imposto dai bit 0 ed 1 della parola di comando. La lunghezza del settore può essere specificata da multipli di 16.

Lettura:

Un'operazione di lettura viene eseguita in 5 passi:
1. Caricamento del registro traccia.
2. Generazione di un «SEEK» (ricerca).
3. Attesa di posizionamento corretto.
4. Trasferimento dati verso il microprocessore sotto controllo interrupt.
5. Verifica che l'operazione è stata eseguita correttamente dopo che è avvenuto il trasferimento dati richiesto.

Scrittura sul Disco:

Questa operazione viene eseguita in 7 passi:
1. Caricamento del registro traccia.
2. Generazione di un ordine «SEEK»
3. Attesa di posizionamento corretto.
4. Emissione di un comando «Write».
5. Caricamento dei dati subito dopo ricezione del segnale di richiesta dati.
7. Verifica degli indicatori di stato «Busy» (occupato) e «CRC - Error».

Altri FDC

Attualmente sono disponibili numerosi altri controllori di floppy-disk, prodotti da Intel, NEC, Motorola e Rockwell. Per esempio nelle Figure 7-23 e 7-24 è riportato un sistema completo impiegante un UPD 372D FDC della NEC. L'interfaccia viene eseguita su un sistema 8080, operante come sistema di controllo ed è in grado di trasmettere dati a qualsiasi calcolatore ricevente.

INTERFACCIA CRT

Un display CRT è molto simile ad un tubo televisivo. (CRT = Cathod-Ray-Tube: tubo a raggi catodici). I dispositivi d'uscita CRT sono molto interessanti per applicazioni d'ufficio e commerciali: essi sono silenziosi e sono in grado di mostrare velocemente intere pagine di dati. Il loro svantaggio è di richiedere una stampante per avere delle copie permanenti.

L'interfacciamento con un CRT comprende sempre un movimento dati a velocità elevata e richiede l'impiego di un DMA. La Fig. 7-25 mostra la configurazione del sistema di base per l'interfacciamento ad un CRT.

Il microprocessore è riportato sulla sinistra dell'illustrazione. Per il rinfresco periodico del CRT viene impiegata la memoria RAM riportata sulla parte destra. Il DMA viene impiegato per il trasferimento automatico di blocchi dalla RAM al CRT. I trasferimenti automatici di blocchi al CRT sono buffered su 2 linee di base. Normalmente sono disponibili due buffers di linea qui indicati dai numeri 1 e 2. Il normale impiego del sistema è il seguente: I dati verranno trasferiti dalla memoria al buffer di linea 1 fino al suo riempimento.

Durante questo periodo di tempo il buffer di linea 2 è assunto pieno. Esso sarà assunto vuoto in una comunicazione di scorrimento con l'uscita video dello schermo. Tipicamente il buffer di linea 2 si vuoterà nello schermo quando quest'ultimo richiede il caricamento dalla RAM al buffer di linea 1. Quando il buffer di linea 2 ridiventa vuoto il multiplexer sulla destra dell'illustrazione commuterà automaticamente sul buffer di linea 1. Progressivamente verrà trasferito sullo schermo il buffer di linea che andrà così vuotandosi.

A questo punto il buffer di linea 2 è vuoto e può essere ricaricato. Ciò avviene automaticamente, sotto il controllo del DMA, dalla memoria e così via.

Fig. 7-23: Interfaccia FDC UPD372D (I).

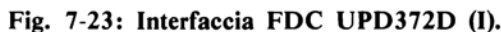
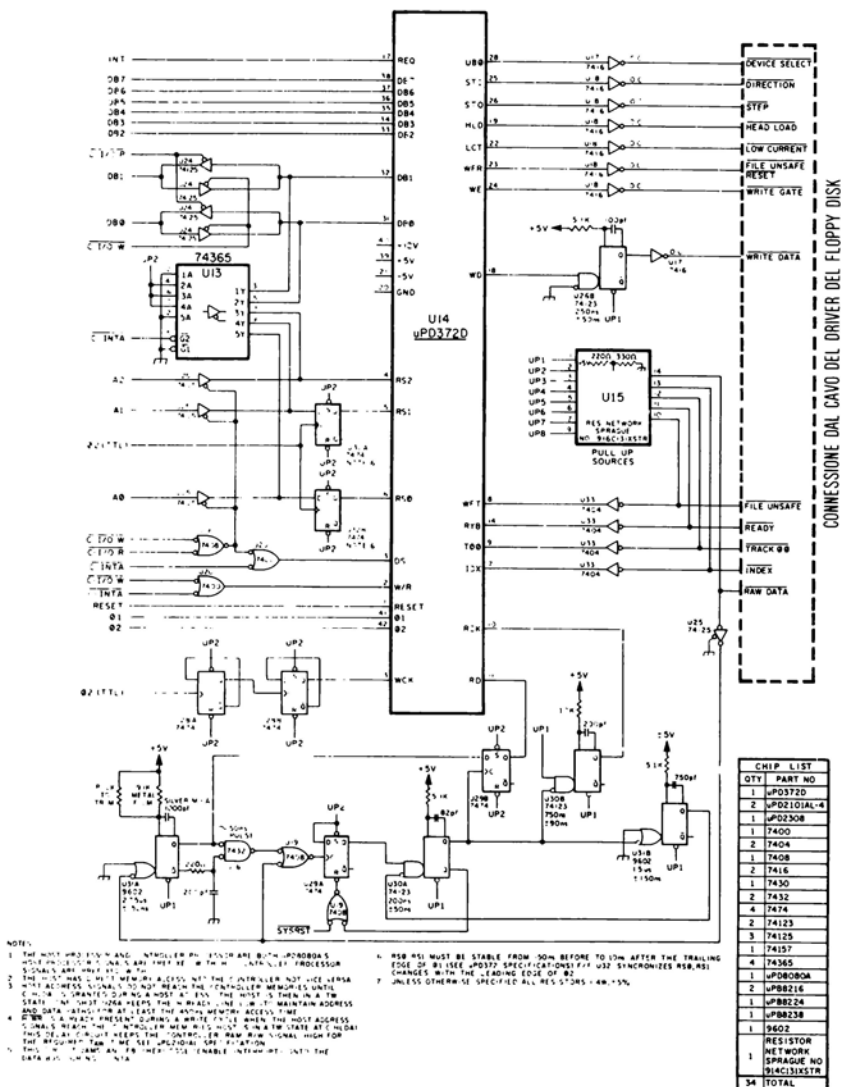



Fig. 7-24: Interfaccia FDC UPD372D (II).

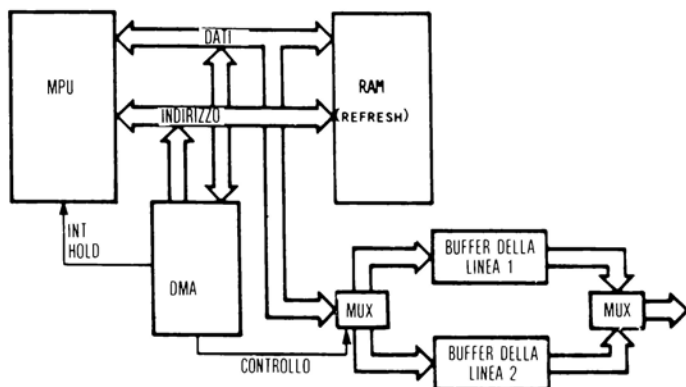



Fig. 7-25: Interfaccia di base del CRT.

In realtà le cose sono molto più complesse. In particolare i bit trasferiti dalla memoria specificano i caratteri in un codice di 8 bit. Questo non è adatto per la visualizzazione dei caratteri sullo schermo. Il metodo normalmente impiegato per mostrare caratteri impiega una *matrice di punti.*

Il codice di carattere ad 8 bit deve essere perciò convertito in una rappresentazione a matrice di punti per essere mostrato sullo schermo. Questo processo di conversione viene eseguito da una ROM oppure da un *generatore di caratteri.* Quest'ultimo viene mostrato in Fig. 7-26. La connessione di un CRTC (controllore CRT), esclusiva del DMA, è riportata in Fig. 7-27. Essa mostra, esplicitamente, il generatore di caratteri richiesto per l'uscita su un registro di scorrimento e su un'uscita video. Assumendo una matrice di punti 5 × 7 per la rappresentazione dei ca-

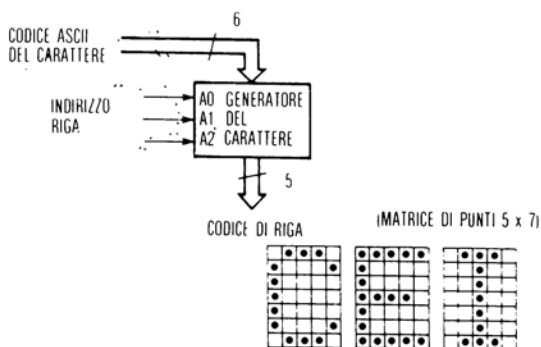



Fig. 7-26: Generazione del carattere.

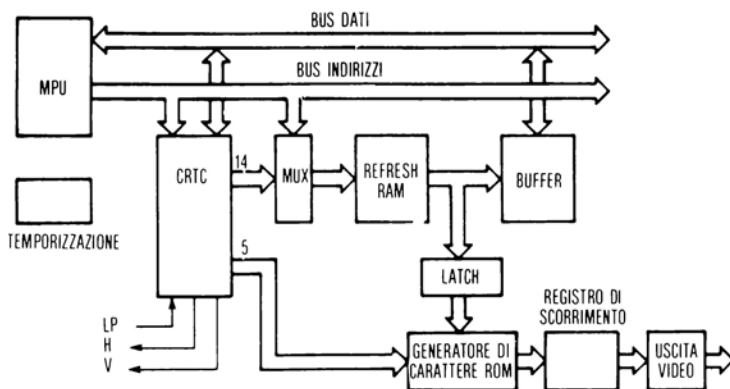



Fig. 7-27: Applicazione del controllore CRT.

ratteri, ogni codice di carattere ad 8 bit originerà un'uscita di 7 × 5 valori (*«punto»* oppure *«non punto»*, cioé nero o bianco). Una ROM di 5 bit può essere utilizzata come generatore di caratteri. Per ogni carattere sono neccessarie sette righe consecutive di punti. Esse saranno presentate all'uscita in successione ordinata. Per specificare una delle sette righe possibili occorre fornire un indirizzo di riga a 3 pin (indicati in Fig. 7-26 con A0, A1 ed A2).

Come ulteriore esempio in Fig. 7-28 è riportata un'interfaccia realizzata con un controllore CRT 8275 della Intel. Il diagramma a blocchi è identico al caso precedente.

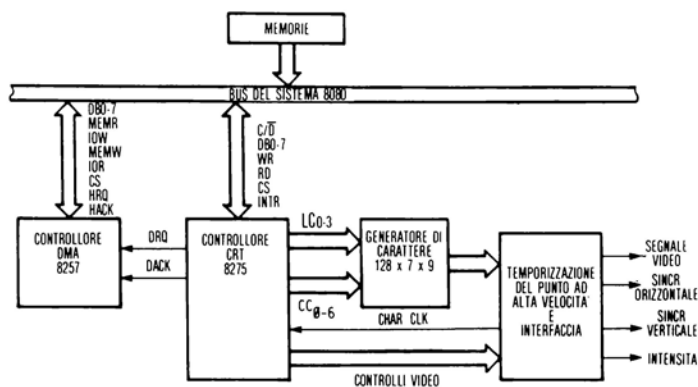



Fig. 7-28: Interfaccia CRT 8275 della Intel.

Le funzioni richieste da un controllore CRT sono numerose. Le funzioni richieste comprendono la capacità di programmare la forma dei caratteri, cioè il numero di punti impiegati per ogni carattere su una linea e la loro distribuzione. Questo viene realizzato da un generatore di caratteri; il numero di righe per il sincronismo; la posizione del cursore (per esempio mediante un quadrato od una forma qualsiasi lampeggiante). Per conservare la traccia della posizione del cursore viene utilizzato un registro speciale. Se viene richiesto l'impiego di una penna luminosa sullo schermo è necessaria la disponibilità di un altro registro.

Inoltre un controllore più potente potrebbe incorporare la maggiore quantità possibile di hardware esterno e quindi il buffers di linea. In un sistema complesso questo controllore potrebbe anche fornire funzioni complesse quali: «*scrolling* » e «*paging*». Lo «*scrolling*» indica il movimento automatico del testo sullo schermo in direzione verticale. Il «*paging*» indica il passaggio da uno schermo pieno ad un altro schermo pieno di n righe.

Le tecniche di interfacciamento standard per l'utilizzazione ottima dei dispositivi d'interfaccia sono già state descritte. Gli scopi di questo libro non consentono dettagli completi ma si ritiene che le tecniche ed i dispositivi presentati in questa sede possano chiarire sostanzialmente le tecniche di interfacciamento. Non si è ancora descritto un ulteriore aspetto dell'interfacciamento: un sistema microprocessore può essere interfacciato da un secondo sistema microprocessore oppure anche da diversi sistemi. Questi sono i sistemi *multi-microprocessore*. Si esamineranno ora le strategie di interconnessione.

SISTEMI MULTI-MICROPROCESSORE

Due o più microprocessori possono essere interconnessi impiegando diverse tecniche. Le due tecniche ottimali per l'interconnessione di microprocessori monolitici sono:

1. Comunicazione per mezzo di memoria.
2. Comunicazione tra registri.

Inoltre esiste una terza tecnica; l'interconnessione diretta al bus, usata molto spesso per le ragioni seguenti. Le due principali strategie delle interconnessioni sono illustrate in Fig. 7-29.

Comunicazione per mezzo di memoria

La strategia della comunicazione per mezzo di memoria è illustrata in Fig. 7-29 in alto. Essa implica l'impiego di una RAM a *due porte*. Una RAM a due porte è una RAM equipaggiata di un'interfaccia speciale che fornisce delle connessioni duali al bus dati ed al bus indirizzi (*due porte*). La priorità di accesso viene stabilita internamente; nel caso di richiesta simultanea una delle due porte ha priorità più elevata determinata per mezzo di hardware.

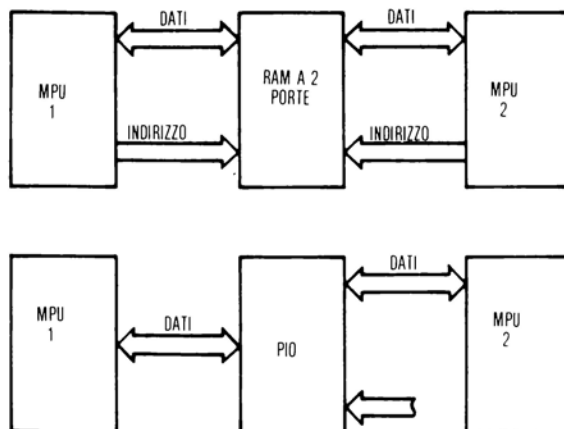



Fig. 7-29: Le 2 strategie di comunicazione.

Il funzionamento di tale sistema è totalmente asincrono. Esso viene denominato sistema *cassetta postale.* Un'area specifica della memoria è riservata alla comunicazione tra microprocessori. Ciascun microprocessore può quindi accedere a quest'area comune ed anche depositare e leggere dati da questa area cassetta postale. Il meccanismo di controllo dell'accesso costituisce un problema ovvio. Un microprocessore deve non modificare i contenuti di un'area di memoria mentre un altro microprocessore li sta leggendo. Questo problema viene risolto da un *«lock»* (serratura) che può essere realizzato per via hardware oppure software. Un lock è un meccanismo per impedire l'accesso ad una zona di memoria ad uno dei processori. Nella memoria possono essere utilizzati dei bit extra dei quali uno è riservato alla protezione di memoria. Ogni volta che tale bit di protezione è posto ad uno, un altro microprocessore non può accedere a tale parola di memoria. Oppure una parola di memoria può essere riservata al controllo di accesso dei dati e contenere di volta in volta i limiti delle zone accessibili ad uno qualsiasi dei due microprocessori.

Questo semplice meccanismo di interconnessione è utilizzato per la suddivisione dei dati in *blocchi.* Esso è dispendioso poichè richiede un'interfaccia speciale a due porte per la memoria. Inoltre esso è relativamente lento perchè comprende il controllo della parola di memoria prima del trasferimento. Queste sono le caratteristiche da un meccanismo di divisione in blocchi.

Comunicazione tra registri

Nella Fig. 7-28 in basso è rappresentato un meccanismo di comunicazione tra registri. In passato la comunicazione tra registri comprendeva l'impiego dei registri effettivi del microprocessore. Ora, con la disponibilità dei chips di interfaccia come il PIO, il registro da suddividere è un registro I/O residente *nel PIO.* In questo caso

metà del PIO è dedicata alla comunicazione del microprocessore. Uno qualsiasi dei microprocessori può quindi depositare una parola di dati all'interno di un PIO mentre l'altro microprocessore può leggerla. Gli interrupts, od i loro equivalenti, possono essere generati da entrambi i microprocessori. Se entrambi i microprocessori devono assumere ruoli equivalenti, lo schema di comunicazione ottimale dovrebbe comprendere l'impiego di due PIO, ciascuno appartenente ad un microprocessore per scopi di manipolazione di interrupts. Questo schema di comunicazione è il più efficiente in un dialogo *orientato alla parola.* Esso potrebbe risultare lento nel caso di trasferimenti di blocchi. Un'altra caratteristica della sua efficienza è la generazione automatica di interrupts ogni volta che una parola dati è depositata all'interno di un PIO.

Entrambi i sistemi precedentemente descritti possono essere impiegati *contemporaneamente.* Se è necessaria una velocità di comunicazione più elevata possibile si può impiegare una connessione diretta al bus. Comunque questo richiede un'ela-

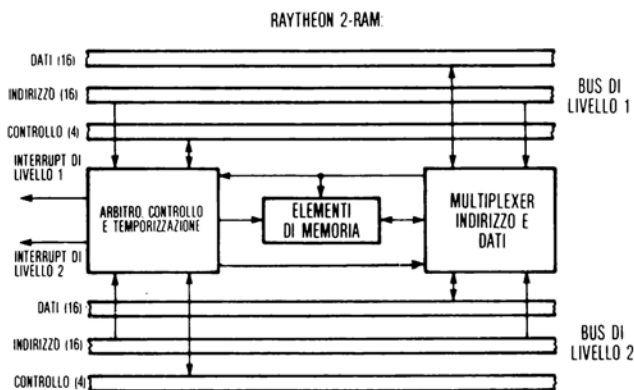



Fig. 7-30: RAM a 2 porte di Raytheon.

borazione indipendente. Attualmente questo è reso possibile dalla disponibilità di microcomputer a chip singolo. La connessione di entrambi i microprocessori sullo stesso bus è possibile solo se uno dei microprocessori è inattivo (sul bus).

Un meccanismo di connessione a più bus chiaramente coinvolge un buffering tra i due bus. Per questa ragione esso è analogo alla comunicazione tra registri realizzata con un PIO. Per questa ragione la vecchia interconnessione «bus comune» è stata sostituita dall'impiego dei PIO. Per esempio la struttura attuale di un sistema multi-processore realizzato da Raytheon per un'applicazione radar è riportata in Fig. 7-30. Si può notare che esso impiega estensivamente *RAM a due porte.* Il processore può essere interconnesso con altre varianti rispetto agli schemi precedenti. Essi possono comunicare anche per via telefonica per mezzo di altre linee di

comunicazione impiegando un dispositivo UART ed un modem. In questa sede sono stati esposti i concetti di base della comunicazione tra registri.

A questo punto l'analisi delle tecniche disponibili, per l'interconnessione di tutti gli elementi logici di un sistema microprocessore, può ritenersi completa. Un fatto nuovo e importante è la disponibilità di un certo numero di bus stadard. Un utente, nel corso dello sviluppo di un sistema comprendente un numero notevole di periferiche, può decidere di selezionare uno dei bus standard esistente, per poter connettere direttamente ad esso le periferiche. È perciò importante analizzare in questa sede i bus standard.

- MASSA	
- DATI TRASMETTITORE	(AL GRUPPO DI COMUNICAZIONE)
- DATI RICEVITORE	(AL GRUPPO DI COMUNICAZIONE)
- RICHIESTA DI INVIO	(AL GRUPPO DI COMUNICAZIONE)
- CLEAR TO SEND	(AL GRUPPO DI COMUNICAZIONE)
- INSIEME DATI PRONTO	(AL GRUPPO DI COMUNICAZIONE)
- TERMINALE DATI PRONTO	(AL GRUPPO DI COMUNICAZIONE)
- INDICATORE AD ANELLO	(AL GRUPPO DI COMUNICAZIONE)
- RICEVUTO RIVELATORE SEGNALE DI LINEA	(AL GRUPPO DI COMUNICAZIONE)
- RIVELATORE QUALITA' SEGNALE	(AL GRUPPO DI COMUNICAZIONE)
- SELETTORE VELOCITA' DATI	(AL GRUPPO DI COMUNICAZIONE)
- SELETTORE VELOCITA' DATI	(AL GRUPPO DI COMUNICAZIONE)
- TEMPORIZZAZIONE TRASMETTITORE	(AL GRUPPO DI COMUNICAZIONE)
- TEMPORIZZAZIONE TRASMETTITORE	(AL GRUPPO DI COMUNICAZIONE)
TEMPORIZZAZIONE RICEVITORE	(AL GRUPPO DI COMUNICAZIONE)
- DATI SECONDARI E RICHIESTE	

Fig. 7-31: Segnali RS232 C.

BUS STANDARD

I quattro bus standard «classici» per i microprocessori sono: RS-232, IEEE 488, CAMAC ed il nuovo bus S-100. Si considerino singolarmente questi quattro bus.

RS-232C

Questa è la classica interfaccia seriale binaria. La velocità di trasmissione risulta 100 ÷ 9600 baud. Si definiscono i livelli di interfaccia standard. Un carattere di stato viene inviato ogni volta che il dispositivo è pronto per accettare un nuovo dato. L'applicazione più frequente è per i display CRT. I segnali tipici del bus RS-232C sono riportati in Fig. 7-31.

IEEE 488

L'emergenza dello standard IEEE 488 risale al 1971 ed è divenuta una realtà ufficiale nel 1975 negli Stati Uniti, quando è stato pubblicato dalla IEEE. Esso viene

anche denominato lo «strumento standard» oppure il «General Purpose - Interface Bus» (GPIB). Alternativamente esso viene chiamato «HPIB» (Hewlett-Packard Interface-Bus) a causa del ruolo essenziale giocato dalla Hewlett-Packard nella promozione di questo standard.

In un connettore singolo da 24 pin sono specificati:

1. Otto linee di massa
2. 16 linee di segnale:
   8 per i dati
   3 per il trasferimento: DAV, NFRD, NDAC
   5 per la direzione del bus: IFC, ATN, REN, SRQ, EOI

Questo bus viene spesso chiamato bus ASCII poichè su di esso non viene specificato il formato ad 8 bit sul bus dati bidirezionale. Si tratta, quasi sempre di un formato ASCII. (Dal 1975 questo bus è anche denominato Standard ANSI MC1.1 - 1975).

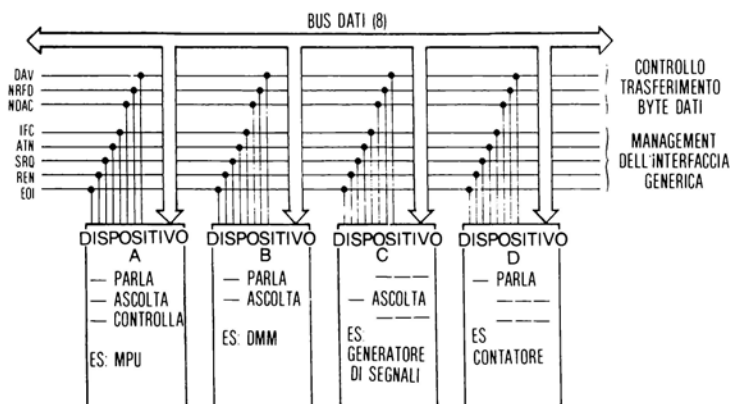



Fig. 7-32: Bus standard per strumenti IEEE 488.

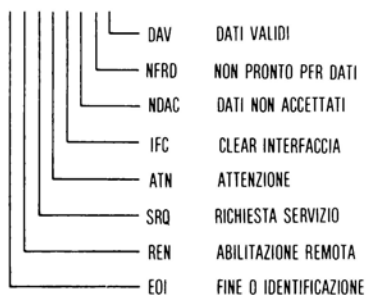



Fig. 7-33: 488: gli 8 segnali di controllo.

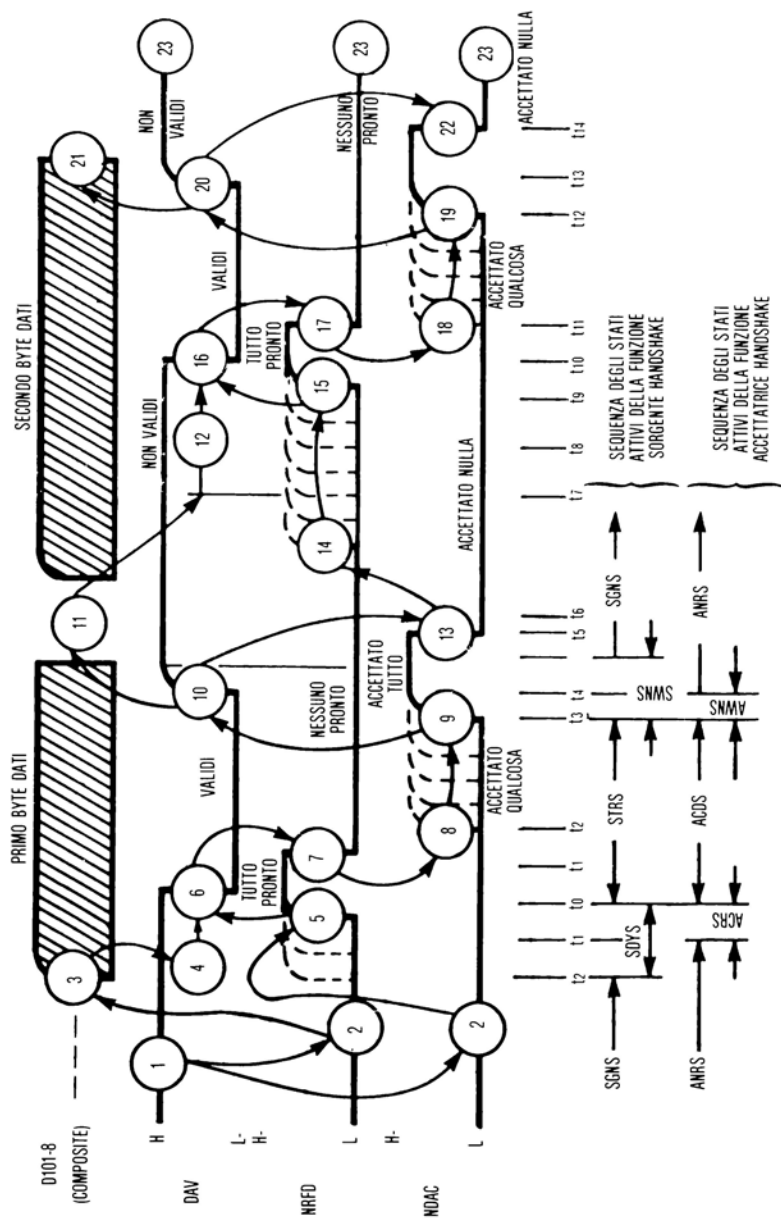



Fig. 7-34: 488: handshake.

La Fig. 7-32 riporta il bus standard IEEE 488. La Fig. 7-33 riporta i segnali di controllo. La Fig. 7-34 riporta il processo di convalida per un trasferimento dati handshake.

Durante un trasferimento dati, la linea ATN si porta al livello basso ed il codice ad 8 bit viene portato sul DIO. I dati sul DIO non hanno un formato unico. Essi possono essere dati, indirizzi oppure istruzioni agli strumenti, misure, comandi universali oppure parole di stato.

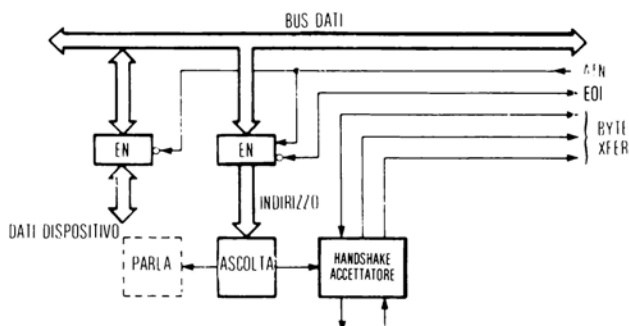



Fig. 7-35: Tipica interfaccia GPIB: ascolta.

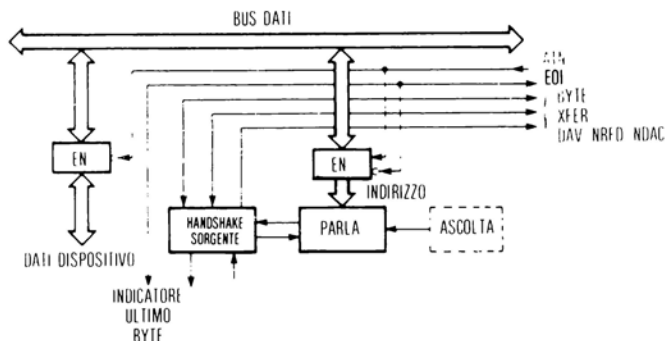



Fig. 7-36: Tipica interfaccia GPIB: parla.

È importante sottolineare che, il fatto che i dati non abbiano un formato unico, costituisce nello stesso tempo una forza ed una debolezza del sistema. Come tali è possibile impiegare la più vasta libertà di interpretazione di questi dati. D'altra parte sarà necessario l'impiego di uno strumento *intelligente* per la rivelazione del flusso di parole ad 8 bit su tale bus. Sarà responsabilità dello strumento connesso a questo bus decodificare la sequenza di parole presentata da esso. Sarà quindi necessaria un'operazione di elaborazione. Le Figure 7-35 e 7-36 riportano due interfacce tipiche al GPIB nella rappresentazione a blocchi (parla ed ascolta).

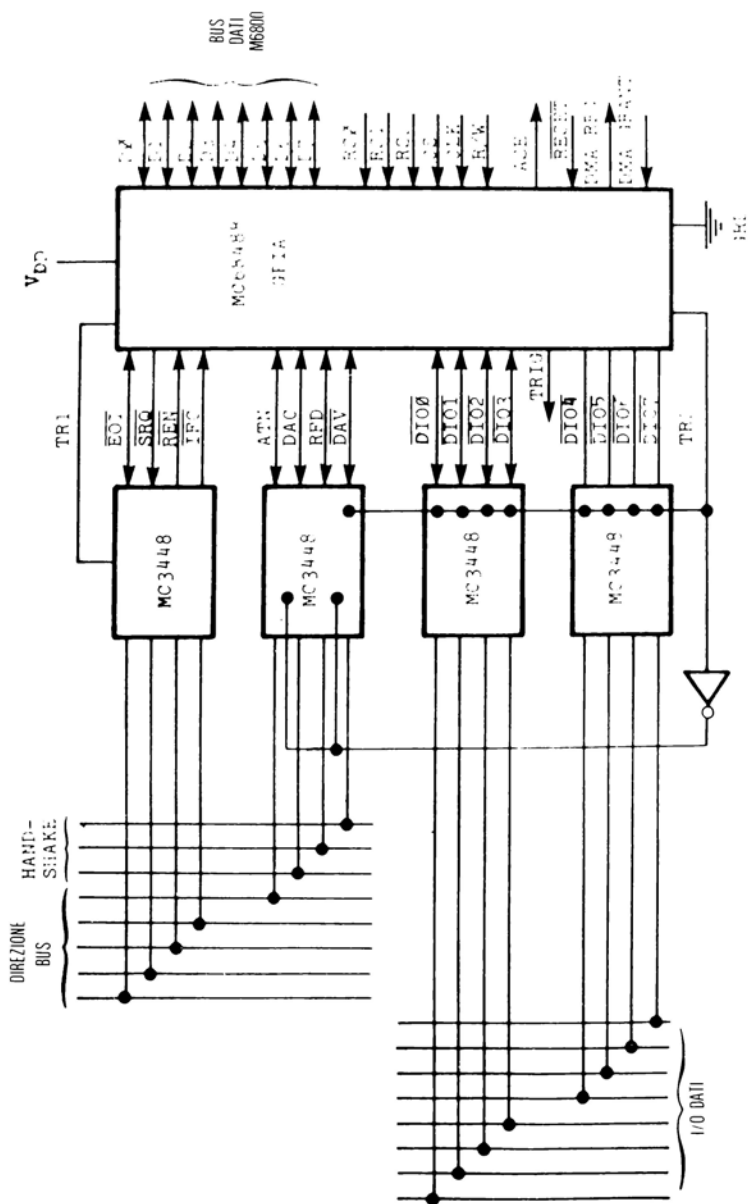



Fig. 7-37: Chip GPIA della Motorola.

I principali limiti fisici del GPIB sono:

- Una velocità di trasferimento massima di 1 Mbyte al secondo.
- Una larghezza di parola di 8 bit (piuttosto che 16, da cui ne risulterebbe un vantaggio nella connessione di strumenti ad un minicomputer).
- Esso richiede che gli strumenti siano vicini (al massimo distanti 2 metri).
- Infine la comprensione dello standard stesso.

D'altra parte il suo vantaggio principale è di consentire una facile connessione di molti strumenti ad un bus universale. La non specificazione del bus richiede un controllo intelligente e complesso al livello degli strumenti connessi al bus stesso. Questo può non avere interesse nel caso di strumenti molto costosi mentre diventa un ostacolo per gli strumenti più semplici.

La Motorola ha annunciato l'introduzione di una specifica interfaccia 488 su chip singolo, denominata interfaccia MC 68488. Questo componente potrebbe divenire una realtà nel prossimo futuro e consentirebbe l'impiego diretto del 6800 con il bus 488. La Fig. 7-37 riporta la connessione di interfaccia.

CAMAC

Il bus CAMAC è lo standard 583 della IEEE. Il bus IEEE 488 costituisce una semplice convenzione di interconnessione hardware. Lo standard CAMAC rispetta questa convenzione e specifica un bus d'interfaccia parallelo con moduli fisici spe-

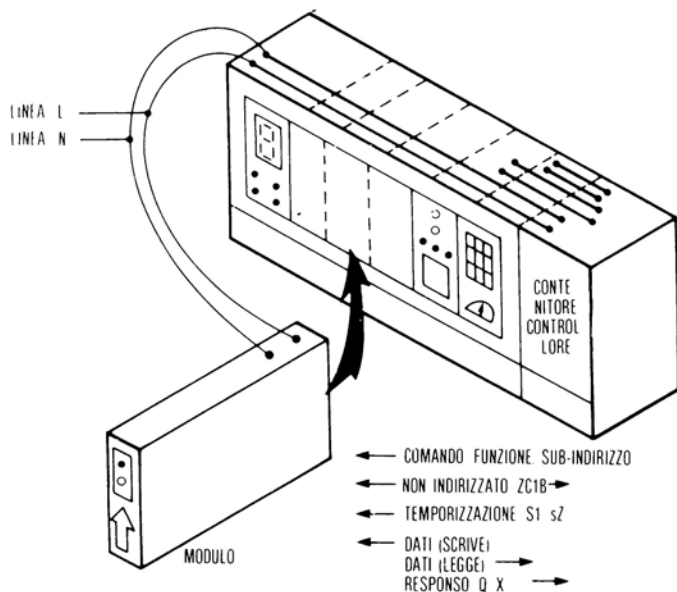



Fig. 7-38: Interfaccia CAMAC standard IEEE 583.

cifici, i *crates* (casse). La parola «crate» è strettamente associata allo standard CAMAC. Ogni modulo viene specificato in dettaglio. Un semplice sistema CAMAC è illustrato in Fig. 7-38. Ogni modulo viene collegato con un connettore da 86 pin. Un crate è un raccoglitore di schede con 25 posizioni. In fondo ad ogni crate risiede un controllore del crate che impiega quasi sempre un microprocessore. Lo standard CAMAC è stato sviluppato per la standardizzazione delle applicazioni nucleari ed ora ha un impiego significativo nel controllo di processi industriali dove è essenziale la modularità.

Il suo svantaggio è di essere uno standard dispendioso a causa del fatto che i moduli sono completamente specificati. Il suo vantaggio è l'intercambiabilità di funzioni sulla base del modulo hardware.

Il bus S - 100

Il bus S-100 viene anche denominato lo standard Altair/Imsai. Esso è divenuto una realtà dopo l'esposizione del mercato hobbystico avvenuta dal 1976 in poi. Nel 1976 due principali produttori di calcolatori per hobby, MITS ed IMSAI, fornirono un bus di cento linee «comune» per la connessione dell'8080 di dispositivi esterni. Ora esso è utilizzato dal 10 per mille degli utenti di sistemi compatibili col bus S-100. Esistono differenze secondarie tra gli standard MITS ed IMSAI, essi sono essenzialmente identici. Il bus S-100 attualmente è divenuto standard di fatto in quanto la maggior parte delle periferiche disponibili nel campo hobbystico sono direttamente interfacciabili con questo bus.

Probabilmente è la prima volta nel settore computing che gli *utenti* impongono uno standard di fatto sull'industria. Non è uno standard ben progettato ma è uno standard esistente e, per questa ragione, avrà una certa stabilità nel tempo.

Il bus S-100 è caratterizzato da una scheda da 5 oppure 10 pollici e da tre livelli di alimentazione: +8V, +18V (16 nella versione IMSAI) e -18V. È molto semplice interfacciare uno Z-80 ad un bus S-100 ed un 6800 può essere ragionevolmente adattato al bus. La Fig. 7-39 riporta i segnali effettivi utilizzati dal bus S-100. La descrizione completa dei segnali è riportata in appendice.

Alcuni problemi correnti del bus S - 100 sono:

- Accoppiamento parassita tra le linee che richiedono isolamento della scheda madre (l'impiego di una pista di massa tra i segnali riduce tale accoppiamento).
- Diverse linee sono ancora non definite e quindi non standardizzate.
- Queste linee vengono impiegate in modi diversi dagli utenti e dai produttori.
- Il bus S-100 è senza alcun dubbio lo standard più impiegato che sia mai stato sviluppato dall'industria dei computer. Questo non può essere ignorato.

Pin	Segnale	Pin	Segnale
1	+8v	51	+8v
2	+16v	52	-16v
3	XRDY	53	SSW DSB
4	VI Ø	54	EXT CLR
5	VI 1	55	*
6	VI 2	56	
7	VI 3	57	
8	VI 4	58	
9	VI 5	59	
10	VI 6	60	
11	VI 7	61	
12		62	
13		63	
14		64	
15		65	
16		66	
17		67	
18	STATUS DSBL	68	MWRITE
19	CCDSBL	69	****
20	**	70	***
21	SS	71	RUN
22	ADDR DSBL	72	PRDY
23	DO DSBL	73	PINT
24	Ø2	74	PHOLD
25	Ø1	75	PRESET
26	PHLDA	76	PSYNC
27	PWAIT	77	PWR
28	PINTE	78	PDBIN
29	A 5	79	A Ø
30	A 4	80	A 1
31	A 3	81	A 2
32	A 15	82	A 6
33	A 12	83	A 7
34	A 9	84	A 8
35	DO 1	85	A 13
36	DO Ø	86	A 14
37	A 10	87	A 11
38	DO 4	88	DO 2
39	DO 5	89	DO 3
40	DO 6	90	DO 7
41	DI 2	91	DI 4
42	DI 3	92	DI 5
43	DI 7	93	DI 6
44	SMI	94	DI 1
45	SOUT	95	DI Ø
46	SINP	96	SINTA
47	SMEMR	97	SWO
48	SHLTA	98	SSTACK
49	CLOCK (2MHz)	99	POC
50	GND	100	GND

* riservato alla massa telaio
** riservato alla memoria non protetta
*** riservato alla memoria protetta
**** riservato allo stato protetto

Fig. 7-39: Bus S-100 (IMSAI).

SOMMARIO

In questo capitolo sono state descritte le principali tecniche di interfacciamento per i più comuni dispositivi d'ingresso-uscita dei microprocessori, assieme ad alcune alternative software ed hardware.

Si sono considerati altri problemi speciali collegati all'interfacciamento come i sistemi multiprocessore ed i bus standard. A questo punto dovrebbe essere semplice l'assemblaggio hardware di un sistema completo. Sono state presentate diverse soluzioni a tutti i problemi significativi.

La tecnica fondamentale che rimane da descrivere è la programmazione. Questo verrà fatto al Capitolo 8.

CAPITOLO 8

PROGRAMMAZIONE DEL MICROPROCESSORE

OBIETTIVO

L'obiettivo di questo capitolo è di presentare un'introduzione chiara alla programmazione. Verranno descritte in questa sede tutte le definizioni principali ed alcune tecniche. Alla fine del capitolo dovrebbero essere chiari i principi coinvolti nella programmazione e la scrittura dei programmi.

Comunque occorre sottolineare che la programmazione non può essere insegnata in un solo capitolo di un libro. Essa richiede una presentazione chiara seguita da applicazioni pratiche. Questo capitolo è rivolto ad un'introduzione chiara dei concetti, problemi e tecniche coinvolte nella programmazione. Si raccomanda vivamente di acquisire un'esperienza diretta.

DEFINIZIONI

Si supponga di dover risolvere un problema di controllo. La soluzione del problema sarà espressa sotto forma di *algoritmo.* Un algoritmo è una specifica passo per passo di una sequenza di operazioni che risolveranno il problema. A questo punto l'algoritmo può essere espresso in qualsiasi forma ed in qualsiasi linguaggio. Per l'impiego di un microprocessore è necessario convertire l'algoritmo nella forma che sarà seguita direttamente dal processore. Si è visto precedentemente che un microprocessore può eseguire soltanto istruzioni *binarie.* Ogni istruzione binaria può essere lunga 1, 2 oppure 3 bytes, dove 1 byte è una sequenza di 8 bit. L'insieme di istruzioni che realizzano l'algoritmo è detto *programma.*

La conversione dell'algoritmo in un *linguaggio* eseguibile dalla macchina è un problema fondamentale. Tale traduzione converte direttamente l'algoritmo in successione di numeri binari. Sfortunatamente i numeri binari non sono facili da impiegare e da ricordare per l'uomo. Per questo motivo sono state escogitate diverse rappresentazioni alternative. Sono stati creati dei linguaggi artificiali nei quali le istruzioni possono essere rappresentate in forma *simbolica.* Essi sono chiamati *linguaggi di programmazione.* Ogni istruzione di un linguaggio di programmazione sarà tradotta da uno speciale programma in una o più istruzioni a livello binario. Esistono due livelli per i linguaggi di programmazione.

Linguaggio - Assembly è una rappresentazione diretta delle istruzioni binarie che possono essere eseguite dal processore. Per esempio «ADD r» per l'8080 è una rappresentazione *mnemonica* di un'istruzione binaria. (Essa somma i contenuti del registro r all'accumulatore). Un programma *assembler* convertirà automaticamente «ADD r» nel codice binario appropriato. Inversamente un *disassembler* convertirà le istruzioni binarie nella loro rappresentazione mnemonica simbolica. L'impiego del linguaggio assembly consente la rappresentazione di tutte le istruzioni binarie in forma simbolica. Poichè ogni istruzione di livello assembly sarà tradotta in *una* istruzione (binaria) di livello macchina, questo è il linguaggio di programmazione più efficiente da un punto di vista di efficienza della macchina. Esso consente la manipolazione diretta dei registri e dei bit all'interno della macchina. Questo è il linguaggio impiegato più frequentemente per qualsiasi applicazione che richiede efficienza di esecuzione.

Sfortunatamente la programmazione in linguaggio assembly è lenta e tediosa, in quanto si devono programmare i trasferimenti di dati ai livelli di bus interno e di registro. A causa di questa inefficienza a livello umano sono stati sviluppati i *linguaggi ad alto livello.* Essi sono più vicini alla rappresentazione funzionale degli algoritmi e sono indipendenti dall'architettura interna del processore. Si è convenuto che non è possibile comunicare direttamente con un computer impiegando il linguaggio ordinario. Il linguaggio umano ha ambiguità di sintassi che devono essere risolte dal contesto o da altre forme di comunicazione non disponibili su un computer. A causa di questa limitazione dei linguaggi umani, occorre impiegare dei linguaggi artificiali aventi una grammatica più semplice e non ambigua. Sono così stati sviluppati numerosi linguaggi ad alto livello. Ciascuno ha una sintassi specifica che lo rende più conveniente per un'assegnata classe di applicazioni quali scientifiche, commerciali ed elaborazione di liste (FORTRAN, COBOL, LISP).

I due linguaggi ad alto livello impiegati più spesso per i microprocessori sono il PL/M ed il BASIC. Il PL/M è derivato dal PL/1 ed è stato introdotto originariamente dalla Intel e quindi dalla maggior parte dei produttori. Il BASIC è stato sviluppato per avere una caratteristica interattiva ed è ora largamente impiegato dai produttori di microprocessori.

Un linguaggio ad alto livello fornisce istruzioni potenti come: «Esegui 24 volte la seguente istruzione» oppure «Finché la variabile n non raggiunge il valore 2024». Non è affatto necessario programmare a livello di registro. I programmi scritti simbolicamente, impiegano i nomi di variabili ed altre strutture dati.

Queste istruzioni ad alto livello devono essere convertite in codice eseguibile dalla macchina. Questa traduzione automatica è realizzata da un programma *compilatore.* Un compilatore traduce ogni istruzione ad alto livello in *molte* istruzioni binarie. Il codice risultante è denominato *codice oggetto* (oppure binario). Poichè la traduzione è automatica vengono fatte molte assunzioni dal computer ed il codice oggetto risultante non è ottimale. In dipendenza dell'efficienza del compilatore il

codice binario risultante sarà da due a cinque volte più voluminoso di quello generato da un programmatore in linguaggio-assembly. Questo origina uno spreco di memoria ed un rallentamento dell'esecuzione del processore. Comunque il vantaggio più evidente è un risparmio del tempo del programmatore, cioè una programmazione più veloce ed efficiente. Questo problema verrà considerato al capitolo successivo dedicato ai sistemi di sviluppo.

Una volta che un programma è stato scritto in linguaggio di programmazione, esso deve essere corretto. L'operazione di *debugging* consiste nella identificazione ed eliminazione degli errori all'interno di un programma. Normalmente il debugging si svolge in cinque fasi:

1. Scrittura del programma.
2. Controllo su carta.
   La correttezza del programma è controllata su carta tracciando a mano l'esecuzione delle istruzioni e controllando manualmente i risultati.
3. Traduzione del programma.
   (Per mezzo dell'assembly o della compilazione a seconda del linguaggio impiegato).
4. Debugging sul processore.
   Il programma viene eseguito sul processore identificando e correggendo gli errori.
5. Esecuzione finale (esente da errori).

Normalmente viene eseguito un passo intermedio tra l'algoritmo ed il programma si tratta del *diagramma di flusso* (flow-chart). Un diagramma di flusso è una

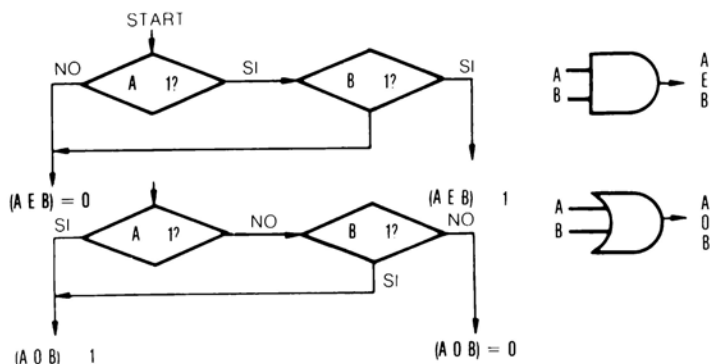



Fig. 8-1: Equivalenza tra logica e programma.

rappresentazione simbolica della sequenza delle operazioni coinvolte nell'algoritmo. I due simboli essenziali impiegati in un diagramma di flusso sono un rettangolo (impiegato per gli ordini, i comandi o le istruzioni da eseguire) ed un rombo (impie-

gato per i test logici). Due o più frecce possono uscire da un rombo a seconda del test. Se il test è binario, cioé se implica una risposta «sì oppure no» dal rombo usciranno due frecce. La Fig. 8-1 riporta degli esempi semplici per l'impiego di questi simboli.

I diagrammi di flusso sono la rappresentazione più frequentemente impiegata per gli algoritmi poichè essi sono indipendenti dal programma e se progettati con cura, possono essere convertiti facilmente in programmi, impiegando qualunque linguaggio di programmazione. A titolo di esempio nel corso del libro saranno mostrati diversi diagrammi di flusso per illustrare algoritmi specifici.

RAPPRESENTAZIONE INTERNA DELL'INFORMAZIONE

All'interno del sistema si devono rappresentare due tipi di informazione: *programma* e *dati*. Le istruzioni di un programma sono sempre rappresentate in una codifica binaria. Esempi specifici sono stati studiati nel caso dell'8080. La codifica delle istruzioni è imposta dal costruttore e non può essere modificata dall'utente. Inoltre l'utente generalmente non avrà problemi di codifica binaria delle istruzioni se programmerà direttamente in esadecimale. La tecnica di programmazione usuale coinvolge la scrittura dei programmi a livello di linguaggio assembly oppure in linguaggio a livello più elevato.

La rappresentazione dei dati che il programma manipolerà costituiscono un problema significativo di rappresentazione dell'informazione. Si devono rappresentare due specie essenziali di dati: *numerici* ed *alfanumerici* (caratteri).

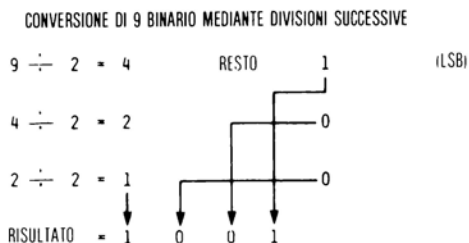



Fig. 8-2: Conversione da decimale a binario.

Rappresentazione dei dati numerici

Otto bit binari possono essere combinati per formare 2^8, cioè 256, combinazioni diverse. Questo non è sufficiente per la rappresentazione dei numeri richiesti ed occorre quindi impiegare una rappresentazione più complessa. Occorre distinguere due casi nella rappresentazione dei numeri: numeri *interi* ed a *virgola mobile* (decimali). Inoltre occorre indicare il *segno* di un numero. La tecnica universale di codi-

fica del segno di un numero consiste nel dedicare il bit di estrema sinistra (MSB) al segno. Uno «0» rappresenta un «+» ed un «1» rappresenta un «—». I bit rimanenti vengono impiegati per la codifica del valore del numero. Verranno ora presentate le quattro rappresentazioni fondamentali.

GRANDEZZA-SEGNO

La rappresentazione *grandezza-segno* è probabilmente la rappresentazione più semplice ed intuitiva. Normalmente il MSB rappresenta il bit del segno. I bit rimanenti rappresentano la grandezza del numero in una rappresentazione binaria diretta. Si è già sottolineato che un singolo byte non è sufficiente per la codifica di numeri con precisione sufficiente. Normalmente viene quindi impiegata una rappresentazione multi-byte, di cui almeno due bytes sono necessari per rappresentare 32K valori (non si dimentichi che un bit viene impiegato per il segno e quindi rimangono soltanto 15 bit utili per una rappresentazione a 2 bytes). Si consideri un esempio della rappresentazione grandezza-segno che mostra anche il suo svantaggio:

«1» è rappresentato (assumendo l'impiego di un solo byte) da: 00000001.
«—2» è rappresentato da: 10000010. («1» è «—», «0000010» è «2»).

Si sommino questi due numeri impiegando le regole dell'ordinaria addizione binaria. Il risultato è 10000011. Questo è «-3» nella rappresentazione grandezza-segno. Le regole dell'addizione binaria diretta non funzionano. (Il risultato corretto è «—1»).

Un vantaggio da considerare nel sistema di rappresentazione numerica non è la facilità di codifica dei numeri, poiché questa verrà eseguita automaticamente dall'assemblatore o dal compilatore. Occorre invece considerare l'efficienza nell'esecuzione di operazioni aritmetiche da parte del processore che porterà ad un'elevata velocità di esecuzione. Si considerino ora delle alternative a questa rappresentazione.

COMPLEMENTO AD UNO

Il *COMPLEMENTO AD UNO* di un numero è il suo complemento matematico. Per esempio 2 sarà rappresentato nella rappresentazione binaria: 00000010. Il complemento ad 1 di 2 in questa rappresentazione è 11111101 («—2»). Ogni «0» è sostituito da «1» e viceversa. Si noterà che il MSB è ancora il bit di segno. Il complemento ad 1 è spesso impiegato nelle grandi unità di elaborazione centrale per migliorare notevolmente l'efficienza del progetto della CPU per aumentare la velocità.

Sfortunatamente esiste ancora un problema: Si sommino 00010110 (+22) ed 11101001 (—22) in complemento ad 1. Il risultato è 11111111. Questo è —127. Questo è lo svantaggio del complemento ad 1. Le regole dell'ordinaria addizione binaria non sono soddisfatte se i segni sono diversi. Si considererà ora una terza rappresentazione, ottimale nel campo dei microprocessori.

COMPLEMENTO A DUE

La notazione in complemento a due è quella più frequentemente impiegata nel settore dei microprocessori per ragioni che verranno immediatamente chiarite. La rappresentazione in complemento a 2 si calcola come segue:

1. Il numero è complementato ad 1.
2. Viene aggiunto «1» alla codifica del numero ottenendo il complemento a 2.

Si osservino alcuni esempi:

«+3» è rappresentato da: «00000011».
Il complemento ad 1 di +3 è: «11111100».
Quindi, aggiungendo 1, si ottiene il suo complemento a 2: «11111101».

Si esegua ora un'addizione di due numeri aventi segni diversi:

```
    00000010 (+2)
+   11111101 (–3)
=   11111111 (–1)
```

Questa è la rappresentazione in complemento a 2 di -1. Quindi è corretto!

DECIMALE	GRANDEZZA—SEGNO	COMPLEMENTO A DUE
-8	- - - -	1 0 0 0
-7	1 1 1 1	1 0 0 1
-6	1 1 1 0	1 0 1 0
-5	1 1 0 1	1 0 1 1
-4	1 1 0 0	1 1 0 0
-3	1 0 1 1	1 1 0 1
-2	1 0 1 0	1 1 1 0
-1	1 0 0 1	1 1 1 1
-0	1 0 0 0	0 0 0 0
+0	0 0 0 0	0 0 0 0
+1	0 0 0 1	0 0 0 1
+2	0 0 1 0	0 0 1 0
+3	0 0 1 1	0 0 1 1
+4	0 1 0 0	0 1 0 0
+5	0 1 0 1	0 1 0 1
+6	0 1 1 0	0 1 1 0
+7	0 1 1 1	0 1 1 1

Fig. 8-3: Complemento a 2.

Il vantaggio fondamentale della rappresentazione in complemento a 2 è quello di non avere problemi con il bit segno quando si eseguono le operazioni di somma e sottrazione. Il risultato è corretto indipendentemente dal segno dei numeri. Questa è una proprietà matematica della rappresentazione in complemento a 2 che può essere verificata con facilità.

Il fatto di non dover verificare esplicitamente il segno di numeri aritmetici durante le addizioni o sottrazioni è alla base di un significativo aumento di velocità di un microprocessore. Per questo motivo la rappresentazione in complemento a 2 viene universalmente adottata dai microprocessori per i numeri con segno. Precedentemente è stata rappresentata una tabella mostrante i decimali equivalenti dei numeri in complemento a 2, impiegando una parola di 8 bit.

NUMERI FRAZIONARI

Fino ad ora sono stati rappresentati soltanto numeri interi. Occorre essere in grado di rappresentare anche numeri FRAZIONARI ovvero decimali. Un limite fondamentale nella rappresentazione di numeri frazionari è dato dal fatto che i programmi aritmetici impiegati per eseguire addizioni, sottrazioni ed altre operazioni aritmetiche, possono funzionare in modo efficiente solo su dati di LUNGHEZZA-FISSA. Cioè la richiesta di una certa efficienza implica la rappresentazione di numeri con un numero fisso di parole. Questo naturalmente limita la precisione dei numeri che possono essere rappresentati nel computer. Eccetto i casi dove è indispensabile una precisione assoluta del risultato, come le applicazioni commerciali, la rappresentazione a *«virgola mobile»* è universalmente adottata per tutti i programmi che richiedono *efficienza di esecuzione*. Nel caso del settore commerciale è stata sviluppata una rappresentazione che risolve il problema della *precisione*. Essa verrà descritta in seguito: è la rappresentazione BCD.

Nella rappresentazione a virgola mobile un numero fisso di bytes viene riservato alla rappresentazione di numeri frazionari. Per esempio si assumerà una rappresentazione a 32 bit ovvero a 4 bytes. Il principio della rappresentazione a virgola mobile è quello di codificare la parola in una parte mantissa ed una parte esponente. Il numero viene quindi rappresentato come: $N = Mx2^E$ dove la M è la mantissa, E l'esponente ed N il numero frazionario.

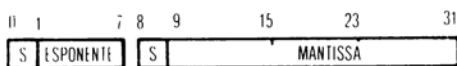



Fig. 8-4: Rappresentazione a virgola mobile.

La *mantissa è normalizzata* in modo da garantire che la rappresentazione conservi il numero massimo di digits della mantissa. Nello schema considerato, sono disponibili 23 bit per rappresentare la mantissa (il bit di estrema sinistra viene impiegato per il segno). Questo corrisponde ad avere i primi K digits per il numero espresso nella comune notazione decimale ed a muovere la virgola decimale a destra del K-esimo digit, moltiplicando poi per un'opportuna potenza di 2 (oppure 10 nella notazione decimale) per codificare la grandezza. Questo garantisce la migliore precisione relativa possibile per il numero di bit impiegati nella rappresentazione. Si

garantisce quindi che il primo bit della mantissa non sia «0». Nella notazione decimale una codifica analoga originerebbe:

$$0{,}1 \leq | M | < 1$$

In binario la mantissa è tale che:

$$2^{-1} \leq | M | < 1$$

Nella rappresentazione considerata, otto bit sono riservati all'esponente con il suo segno e 32 bit sono riservati alla mantissa con il suo segno. Questa è una rappresentazione molto comune. Se è richiesta maggiore precisione occorre considerare un numero maggiore di bit. Naturalmente le regole delle operazioni su numeri rappresentati in virgola mobile sono più complesse di quelle impiegate per le operazioni dirette su numeri in complemento a 2. Occorre scrivere delle *routines in virgola mobile*. Lo scopo di questi programmi è ulteriore a questo libro introduttivo. Brevemente, nel caso di addizione e sottrazione, i numeri devono essere normalizzati in modo da rendere uguali le mantisse e sommando gli esponenti e quindi normalizzando i risultati. Naturalmente l'esecuzione di routines in virgola mobile sarà molto lenta rispetto all'esecuzione di routines artimetiche convenzionali.

Se l'esecuzione di routines in virgola mobile risulta troppo lenta per una data applicazione, esistono dei nuovi chips che eseguiranno l'aritmetica in virgola mobile e si connetteranno direttamente ai bus standard.

DECIMALE CODIFICATO BINARIO (BCD)

La rappresentazione in virgola mobile è normalmente adatta per qualsiasi tipo di problema dove è sufficiente una precisione *elevata* del risultato, piuttosto che una precisione *assoluta*. Sfortunatamente nelle applicazioni di tipo commerciale non si possono tollerare errori. Per esempio la contabilità di una compagnia non può tollerare errori «piccoli di qualche per cento» perchè possono ammontare a diverse centinaia di migliaia di dollari. Questo è chiaramente inaccettabile. Occorre quindi escogitare una nuova rappresentazione che garantisca la precisione su ogni cifra del risultato. Sfortunatamente questo può essere ottenuto soltanto impiegando un formato variabile per la codifica dei numeri. Questo richiede diversi tipi di elaborazione seriale.

Il principio del BCD è semplice. Si devono codificare dieci cifre da 0 a 9. La codifica di dieci cifre richiede l'impiego di 4 bit (tre bit codificherebbero solo *otto* combinazioni). Quattro bit ne codificano 16. Quindi per codificare le dieci cifre decimali sono impiegati quattro bit. Per esempio «1» è rappresentato da 0001 e 9 da 1001. Sfortunatamente quattro bit generano 16 combinazioni possibili e le combinazioni da 1010 ad 1111 non vengono impiegate, cioè *non sono consentite* nella codifica BCD. Ne risulta una complicazione delle operazioni aritmetiche. Si provi a

sommare i digits BCD:

0001 («1») + 0011 («3») = 0100 («4»)

Si ottiene il risultato corretto. Si provi ancora:

1000 («8») + 1000 («8») = 10000.

	2^3	2^2	2^1	2^0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

FORMATO BCD	2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0
29:	0	0	1	0	1	0	0	1
	2				9			

Fig. 8-5: Formato BCD.

Questo è un «1» seguito da «0» in BCD, cioè 10. Il risultato *non è corretto.* Il problema è ovvio. Quando si utilizzano le regole dell'ordinaria addizione binaria i sei codici non consentiti da «1010» ad «1111» devono «essere eliminati». Questo equivale a dire che «si deve aggiungere 6 al risultato», ogni volta che quest'ultimo cade in un codice non consentito.

Si verifichi: 10000 + 0110 («6») = 10110 (aggiunta di 6).

Questo è «1» seguito da «6», cioè «16» in BCD. Il risultato è corretto. Quindi ogni volta che si impiega la rappresentazione BCD occorre sommare 6 al risultato nei casi in cui l'addizione cadrebbe nei codici binari non consentiti. Un'istruzione «DAA», eseguita dopo un'addizione, opererà automaticamente la regolazione richiesta. È interessante notare che virtualmente tutti i microprocessori sono dotati di questa istruzione. Questo è sorprendente e complica notevolmente il progetto dell'unità di elaborazione. La ragione è storica. Si è visto che i microprocessori sono derivati dall'evoluzione dei calcolatori tascabili. Molti calcolatori tascabili ed altri calcolatori da tavolo utilizzavano la notazione BCD per applicazioni contabili. Ne è risultato che questo funzionamento speciale è stato semplicemente trasferito nel progetto del microprocessore.

Non esistono altri problemi. La sottrazione in BCD è ancora complessa. La sottrazione deve essere eseguita impiegando la seguente regola matematica: «Al primo

numero si somma il completamento a 10 del secondo». Sfortunatamente la maggior parte di microprocessori non è equipaggiata di sottrazione BCD cosicché occorre eseguire esplicitamente il complemento a 9. Naturalmente in questo caso è necessario impiegare alla fine l'istruzione DAA.

Infine sorge un altro problema pratico. Un digit BCD è codificato in 4 bit. Un numero decimale richiederà un certo *numero di nibbles* per la sua rappresentazione (1 nibble = 4 bit). Tipicamente il primo nibble della sua rappresentazione sarà il bit del segno e tutti i nibbles seguenti rappresenteranno i digits BCD. L'esecuzione dell'addizione o sottrazione risulterebbe notevolmente inefficiente se si operasse su un nibble alla volta. Possibilmente è desiderabile operare su un byte intero. Questo è quanto avviene in pratica. Comunque, all'interno di un risultato ad 8 bit, si può generare un riporto da un nibble al successivo. Questo riporto deve essere intercettato per correggere il risultato. Questo implica la disponibilità di uno speciale bit flag che rivela un riporto dal bit 3 al bit 4. Si tratta di *half-carry* (H oppure AC) già introdotto al Capitolo 2. Controllando questo bit il programmatore sarà in grado di eseguire un'azione correttiva appropriata ogni volta che l'addizione di due nibbles origina un bit sommato al nibble di sinistra di un registro.

Rappresentazione di dati alfanumerici

La codifica di caratteri è richiesta per la rappresentazione del testo oppure per altri simboli speciali. Normalmente sono codificati almeno 82 simboli: 52 simboli sono necessari per rappresentare le maiuscole e minuscole delle 26 lettere dell'alfabeto, dieci simboli sono richiesti per le dieci cifre decimali e poi normalmente sono richiesti 20 caratteri speciali, come +, –, !, ?, ecc. Occorre impiegare una codifica binaria a 7 bit. 7 bit consentono 128 combinazioni possibili e sono sufficienti per tutti gli insiemi di caratteri. Poiché i microprocessori sono strutturati in bytes di 8 bit, i caratteri sono codificati nel formato ad 8 bit. L'ottavo bit è normalmente impiegato per contenere l'informazione di parità, normalmente impiegata per migliorare l'affidabilità delle trasmissioni. Per facilitare lo scambio di informazioni si sono sviluppati due standard impiegati universalmente. Essi sono lo standard ASCII (American Standard Code for Information Interchange), impiegato dalla grande maggioranza di tutti gli utenti di microprocessori (ed anche di altri computer) ed il codice EBCDIC, impiegato dall'IBM. Il codice EBCDIC è essenzialmente analogo a quello ASCII ma modifica la sequenza secondo cui sono codificati i caratteri, consentendo di migliorare l'efficienza durante i confronti di una classe di caratteri della sequenza. Per tutti gli scopi pratici in questa sede è di rilievo soltanto il codice ASCII a meno che non sia richiesta l'interfaccia diretta ai prodotti IBM.

Sorge a questo punto un problema interessante: un codice identico di 8 bit deve essere impiegato, per la codifica di dati binari, oppure un carattere, oppure anche un'istruzione. Come fa il processore a sapere cosa esso rappresenta? La chiave è il fatto che queste rappresentazioni sono impiegate in un contesto ristretto e *specifico*. Le *istruzioni* sono impiegate in un'area di programma della memoria da dove

saranno caricate automaticamente all'interno del microprocessore per l'esecuzione. Ogni volta che i *dati* sono manipolati dal processore, è responsabilità del programmatore assicurarsi che essi siano manipolati («interpretati») in modo corretto. Si può decidere sotto la propria responsabilità di sommare *caratteri!* Il microprocessore eseguirà l'addizione richiesta senza curarsi del significato del risultato. Impiegando un assembler oppure un linguaggio ad alto livello, esiste la possibilità di codificare automaticamente i numeri ed i caratteri nella loro rappresentazione binaria. Comunque è ancora responsabilità del programmatore eseguire le operazioni «corrette» sui dati da impiegare.

RAPPRESENTAZIONE ESTERNA DELL'INFORMAZIONE

Per la rappresentazione esterna dell'informazione si impiegano tre metodi fondamentali:

1. Idealmente il sistema dovrebbe mostrare l'informazione nel formato più conveniente. Questo implica la rappresentazione *simbolica*. Infatti impiegando delle tecniche sofisticate di debugging sarà possibile mostrare i risultati o i contenuti di registri in forma simbolica. I caratteri dovrebbero quindi essere rappresentati come tali ed i numeri con la loro rappresentazione effettiva, decimale o di altro tipo. Le istruzioni potrebbero essere elencate come mnemonici.

2. Quando queste tecniche non esistono, oppure quando è necessario considerare i dati in modo più preciso, si può ricorrere ad una rappresentazione *binaria*. Ovviamente questa è la rappresentazione meno conveniente per l'impiego.

BINARIO				ESADECIMALE
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

Fig. 8-6: Esadecimale.

3. La rappresentazione attualmente impiegata in modo universale nel mondo dei microprocessori è quella *esadecimale.* Essa è molto simile alla rappresentazione binaria ma rappresenta un significativo miglioramento rispetto ad essa. Nella rappresentazione esadecimale un gruppo di quattro bit binari viene codificato in un digit esadecimale. Come si può dedurre dal nome, la rappresentazione esadecimale codifica 16 combinazioni in un unico simbolo. I digits da 0 a 9 sono utilizzati per rappresentare il loro esatto equivalente binario. Sono necessari altri sei digits. Essi sono semplicemente le lettere dell'alfabeto. Le lettere da A ad F sono impiegate per rappresentare 1010 fino ad 1111 (vedere Fig. 8-6). Il vantaggio fondamentale del codice esadecimale è che gli otto digits binari possono essere codificati con due simboli esadecimali. Quindi l'interfaccia umana è molto più semplice; è molto più facile manipolare un codice di due simboli che una codifica binaria di 8 bit.

Esiste un'ulteriore ragione del travolgente successo del codice esadecimale nel settore dei microprocessori: La tastiera esadecimale deve essere uno dei mezzi d'ingresso più a buon mercato per un microprocessore. Ogni tasto può essere contrassegnato da un simbolo esadecimale. Per caricare una parola nella macchina è quindi necessario premere solo due tasti mentre verranno caricati otto bit.

La conversione da esadecimale a binario viene normalmente eseguita in software.

D'altra parte i LED sono i dispositivi d'uscita più a buon mercato e possono essere impiegati per rappresentare numeri esadecimali. Qualsiasi sistema microprocessore conveniente dovrebbe essere dotato di almeno sei LED.

– Quattro LED mostrano i 16 bit di un indirizzo.
– Due LED mostrano i contenuti ad 8 bit di un registro.

DOMANDA: *Qual'è l'equivalente esadecimale di 0011 0010?*

RISPOSTA: *32.*

DOMANDA: *Qual'è l'equivalente binario di F5?*

RISPOSTA: *Lasciata al lettore.*

FORMATI DELLE ISTRUZIONI

Si è visto che le istruzioni sono ordini impartiti al microprocessore, che vengono codificati dalla sua unità di controllo. Per microprocessori convenzionali ad 8 bit, il formato è 1, 2 oppure 3 bytes. Più breve è l'istruzione e più velocemente verrà eseguita. Per le più lunghe occorrerà molto tempo per il prelievo dei bytes dalla memoria. È possibile classificare i tipi di istruzioni seguendo diversi criteri.

Un criterio ovvio è la natura dell'ordine da eseguire. Si possono quindi distinguere operazioni: *logiche, aritmetiche* e di *controllo.*

È anche possibile distinguere le istruzioni in funzione dell'elemento su cui operano. Si distinguono quindi: operazioni di *tipo registro,* di *tipo memoria* e di *ingresso-uscita.*

La codifica delle istruzioni in bit non è importante. Essa è diversa per ogni microprocessore ed è fissata una volta per tutte dal costruttore.

Per valutare la potenza di un insieme di istruzioni è necessario capire le sue capacità. Le capacità aritmetiche e logiche sono normalmente chiare ed abbastanza simili da un microprocessore ad un altro. Si focalizzerà qui un'altra caratteristica «nascosta» di un set di istruzioni, ed in particolare, le sue caratteristiche di *indirizzamento.*

In generale i *campi* in cui un'istruzione può essere classificata possono essere il *codice operativo* ed un opzionale campo *letterale.*

Per motivi di efficienza, il *codice operativo* (ovvero *opcode*) normalmente occupa i primi otto bit dell'istruzione. In senso stretto, il codice operativo dovrebbe far riferimento solo all'*operazione* specificata e non comprendere qualsiasi specifica di registro. Questo è il caso dei minicomputer e computer tradizionali. È prassi comune, nel caso dei microprocessori, chiamare «codice operativo» la specifica dell'operazione comprendendo la definizione di qualsiasi registro che può comparire entro il campo.

Ogni volta che un'istruzione impiega due o più parole, quelle successive alla prima parola vengono chiamate campo *letterale.* Il campo letterale contiene un *operando* (dati) oppure un *indirizzo.*

Per il codice operativo di un microprocessore «standard» sono disponibili solo otto bit. Questo indica che potrebbero essere fornite al massimo 256 istruzioni diverse. D'altra parte devono essere forniti diversi modi di indirizzamento. Normalmente almeno due bit sono riservati a questa funzione, questo lascia solo sei bit per il codice operativo effettivo, cioè solo 64 possibili istruzioni diverse. *Questa è la ragione per cui nessun microprocessore ad 8 bit ha più di 64 istruzioni diverse.* I costruttori spesso elencano un gran numero di istruzioni poichè differenziano tra i diversi modi di indirizzamento.

Tecniche di indirizzamento

L'indirizzamento di una parola nella memoria è un modo di specificare come essa può essere recuperata. Per avere efficienza nella manipolazione di diverse strutture dati, devono essere disponibili diversi modi di indirizzamento. Spesso i microprocessori forniranno solo un sottinsieme delle possibilità che verranno descritte di seguito. Conseguentemente l'accesso ai dati potrebbe risultare richiedere un gran

numero di operazioni. I principali modi di indirizzamento sono illustrati in Fig. 8-7 e successive.

Distingueremo le sette tecniche di indirizzamento più importanti. Combinazioni di queste tecniche possono anche essere eseguite.

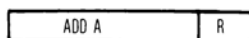



Fig. 8-7: Indirizzamento implicito — Esempio: ADD A, R.

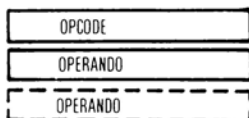



Fig. 8-8: Indirizzamento immediato.

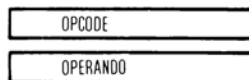



Fig. 8-9: Indirizzamento diretto (indirizzo ad 8 bit).

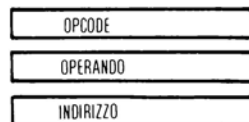



Fig. 8-10: Indirizzamento esteso (indirizzo a 16 bit).

INDIRIZZAMENTO IMPLICITO

Indirizzamento implicito significa che il codice del registro da utilizzare da parte di una certa istruzione non appare esplicitamente nel codice operativo. L'indirizzamento implicito è normalmente impiegato per l'accumulatore o per altri registri destinati a scopi speciali come il PC oppure SP. Un esempio è l'istruzione ADD r che è stata appena descritta. Essa significa «SOMMA (ADD) il registro r all'accumulatore e lascia il risultato nell'accumulatore». Questa istruzione non fa specifico riferimento all'accumulatore. Esso viene indirizzato implicitamente. Naturalmente il riferimento all'accumulatore è *contenuto* nel codice operativo. In questo modo non è richiesto un codice di 3 bit. Si ottiene così un miglioramento dell'efficienza di allocazione dei codici operativi. Se accumulatore fosse stato specificato esplicitamente come il registro r, dovevano essere utilizzati *6* bit. Quindi solo due bit potrebbero essere lasciati al codice operativo. Questo è chiaramente inaccettabile.

Il vantaggio della notazione implicita dovrebbe essere chiaro da questo esempio. Essa origina una notazione più breve ed, in particolare, rende possibile l'impiego di un'istruzione di 8 bit, più efficiente da eseguire su un sistema microprocessore. Si potrebbe dire che l'indirizzamento implicito dovrebbe quindi essere utilizzato per il riferimento di qualsiasi registro interno alla macchina. Questo è fattibile ma porterebbe, a sua volta ad un'elevata complessità di livello di decodificatore interno. Questo aumento di complessità è ancora inaccettabile nel caso dei microprocessori. Si devono quindi realizzare alcuni compromessi tra l'indirizzamento *implicito* ed *esplicito.* (L'indirizzamento esplicito impiegando un codice di 3 bit per otto registri è molto semplice da codificare internamente all'interno del microprocessore e può essere portato direttamente al registro multiplexer). La Fig. 8-7 mostra l'indirizzamento implicito.

INDIRIZZAMENTO IMMEDIATO

L'indirizzamento immediato fa riferimento al fatto che l'operando segue immediatamente il codice operativo. L'operando può essere a 8 o 16 bit. L'istruzione completa può essere perciò di due o tre parole. Questo è illustrato in Fig. 8-8. L'operando letterale che appare nella seconda, ed eventualmente terza, parola può essere un dato oppure un indirizzo. Per esempio esso sarà caricato in un registro interno, oppure sommato ai contenuti di un altro registro interno.

INDIRIZZAMENTO DIRETTO

Un'istruzione di *diramazione*, che è un salto ad una locazione specifica della memoria, potrebbe richiedere normalmente tre bytes: un byte per il codice operativo e due bytes per un indirizzo a 16 bit. L'esecuzione di tali diramazioni è lenta se richiede tre accessi di memoria. Per migliorare la velocità delle operazioni di diramazione, specie nel caso in cui il tempo di azione è critico (come nella risposta ad un interrupt) viene normalmente utilizzato un meccanismo di *indirizzamento diretto* che consente la diramazione alle locazioni da 0 a 255. Queste 256 locazioni possono essere codificate mediante un codice di 8 bit. Per la diramazione ad una di queste locazioni può quindi essere utilizzata un'istruzione di 2 bytes. La Fig. 8-9 illustra questo metodo di indirizzamento. Esso viene fornito esplicitamente nel caso dell'8080 ma con alcune restrizioni (si può saltare solo ad un multiplo di 64 parole? Comunque l'istruzione RST nel caso dell'8080 è realizzata con un solo byte!).

INDIRIZZAMENTO ESTESO O NORMALE

Questo indirizzamento è illustrato in Fig. 8-10. Si tratta di un normale indirizzamento di memoria, impiegante un formato di 3 parole: la prima parola contiene il codice operativo e le altre due contengono l'indirizzo a 16 bit. Esso è utilizzato per qualsiasi tipo di diramazione normale o di operazione di salto. Il termine «esteso» è utilizzato da alcuni costruttori semplicemente per differenziarlo dall'indirizzamento «diretto» dove è consentito soltanto un indirizzo di salto di 8 bit. Comunque questo

è più un termine commerciale, piuttosto che una realtà tecnica e probabilmente questo indirizzamento dovrebbe essere qualificato come indirizzamento di memoria regolare.

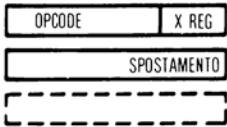



Fig. 8-11: Indirizzamento immediato (I).

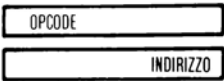



Fig. 8-12: Indirizzamento relativo: PC ± 127.

INDIRIZZAMENTO INDICIZZATO

Nella maggior parte dei programmi è necessario far riferimento al contenuto di una tabella memorizzata nella memoria. Una tabella è semplicemente un gruppo di n parole memorizzate in sequenza. Può essere necessario fare spesso riferimento ad un ingresso ad una tabella. «Si faccia riferimento al terzo ingresso» e quindi «si faccia riferimento al settantaduesimo ingresso». Esiste un modo di indirizzamento specifico che facilita tale accesso a tabelle.

I contenuti di un registro IX, o registro indice, sono impiegati per conservare l'indirizzo di partenza di una tabella nella memoria. Un'istruzione di indirizzamen-

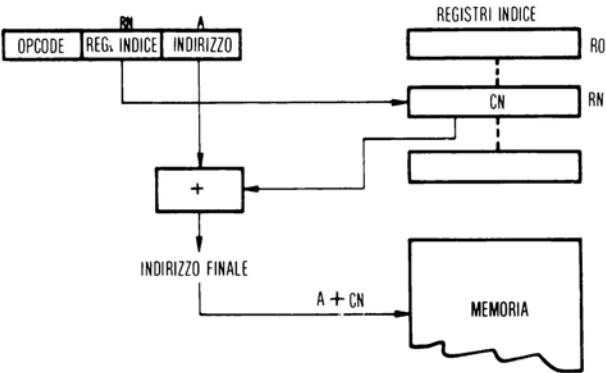



Fig. 8-13: Indirizzamento indicizzato (II).

to indicizzato conterrà un campo di spostamento che verrà sommato automaticamente ai contenuti del registro indice. La Fig. 8-13 illustra questo processo. Spesso il campo di spostamento è limitato ad otto bit cosicchè l'istruzione di indirizzamento indicizzato può essere specificata in soli due bytes. In questo caso la lunghezza massima della tabella può essere di sole 256 parole. Naturalmente un metodo alternativo è quello di specificare l'indirizzo di partenza della tabella entro le prime 256 locazioni e considerare i contenuti del registro indice come un campo di spostamento. Idealmente il campo di spostamento dovrebbe essere di 16 bit così da poter essere scambiato con il registro indice. La differenza fondamentale tra memoria e registri è la seguente: le istruzioni, dopo che sono state immagazzinate su ROM, non possono essere cambiate. Altre volte l'indirizzo di partenza della tabella sarà fisso mentre lo spostamento cambierà dinamicamente. Nel caso dei microprocessori ci si può perciò aspettare che l'indirizzo di partenza della tabella dovrebbe essere posizionato nel campo di istruzione mentre lo spostamento dovrà risiedere nel registro indice. La disponibilità di indirizzo indice dovrebbe essere valutata sulla base delle precedenti considerazioni.

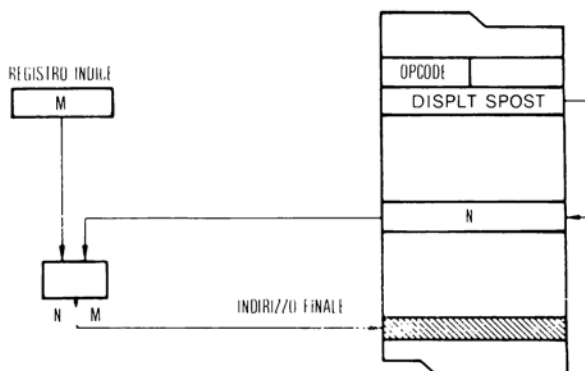



Fig. 8-14: Pre e post-indicizzazione.

Una variazione della tecnica usuale viene mostrata in Fig. 8-14; si tratta dei modi, «pre-indicizzato» e «post-indicizzato». Nel modo post-indicizzato i contenuti di un breve campo di spostamento sono interpretati come indirizzo di uno spostamento finale nella memoria. Questo potrebbe essere considerato «indicizzato-indiretto».

In un processore possono essere disponibili diversi registri indice e questa è certamente una caratteristica apprezzabile. In questo caso l'istruzione deve contenere un puntatore al registro indice da impiegare. Questo è illustrato in Fig. 8-13 dove il campo del registro indice seleziona uno dei registri indicizzabili. La parte rimanente dell'accesso di memoria viene eseguita, come al solito, sommando i contenuti del

campo di spostamento ai contenuti del registro indice selezionato. Questo origina un indirizzo finale che viene utilizzato per l'accesso alla memoria.

INDIRIZZAMENTO RELATIVO

L'indirizzamento relativo è progettato per facilitare i cicli (loops) brevi, cioé i salti entro una distanza di 256 parole. Tali salti possono essere specificati da un'istruzione di due parole, mostrata in Fig. 8-12.

Il vantaggio dell'indirizzamento relativo è ovvio. Esso consente l'esecuzione di un salto veloce in due istruzioni. Un salto generalizzato richiede tre bytes. Esso viene chiamato «relativo» in quanto implica un salto all'indirizzo corrente più lo spostamento specificato. Poichè lo spostamento potrebbe essere positivo o negativo. Questo implica un salto in avanti fino a + 126 locazioni ed un salto indietro fino a −127 locazioni. L'indirizzo relativo generato da quest'istruzione è uguale a PC più lo spostamento assegnato. Se un bit viene impiegato per il segno ne risulta che per lo spostamento rimangono solo sette bit e quindi è limitata a ±127.

L'indirizzamento relativo può essere molto importante per l'efficienza di routines brevi, che richiedono l'esecuzione di un ciclo breve per molte volte, quali sono le routines aritmetiche oppure i trasferimenti di blocchi di dati.

INDIRIZZAMENTO INDIRETTO

L'indirizzamento indiretto viene illustrato in Fig. 8-15. Un'istruzione specifica un indirizzo nel modo indiretto. Questo normalmente viene indicato in linguaggio assembly in modo simbolico con un «*» aggiunto all'istruzione.

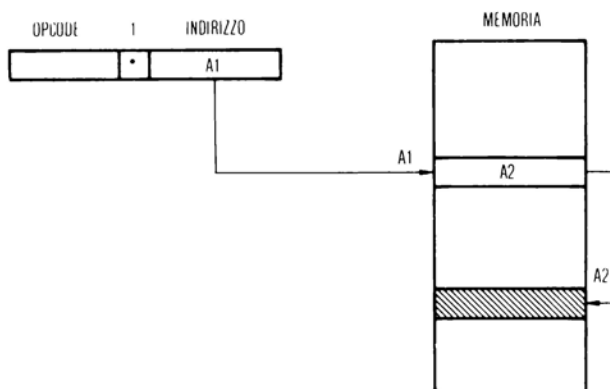



Fig. 8-15: Indirizzamento indiretto.

Indirizzamento indiretto significa «vai all'indirizzo di memoria specificato da A1 e preleva i suoi contenuti (A2). Quindi impiega A2 come indirizzo finale di destina-

zione». Questo implica a sua volta, che i «contenuti» di A1 costituiscano una parola di 16 bit. Essi richiederanno quindi due bytes nella memoria.

L'indirizzamento indiretto viene universalmente impiegato per suddividere l'informazione tra utenti diversi di diversi programmi. Questo equivale a nascondere una chiave in un sottovaso. La posizione della chiave è nota. Essa si troverà nella locazione di memoria A1. Ogni programma che desidera accedere alla parola residente alla locazione A2 (che può essere variata spesso) deve accedere ad una locazione fissa di memoria A1 e trovarvi il suo indirizzo (appunto la sua chiave).

Tipicamente quando vengono eseguiti diversi programmi, i dati risiedono in locazioni non note a priori all'interno della memoria RAM. Ogni volta che vengono impiegati dati di lunghezza variabile, le tabelle memorizzate nella RAM devono essere in accordo con i puntatori di queste strutture dati. Qualsiasi routine, o qualsiasi programma, che richiede l'accesso a questi dati preleverà l'indirizzo effettivo dei dati in questa tabella di puntatori. Si può quindi accedere a dati in modo efficiente impiegando l'indirizzamento indiretto che preleva automaticamente i dati i cui indirizzi sono riportati in corrispondenza di un ingresso specifico della tabella.

Poichè l'accesso a dati suddivisi viene normalmente realizzato per mezzo di tabelle piuttosto che di una singola locazione di memoria, l'indirizzamento indiretto è più efficiente quando è accoppiato all'indirizzamento indicizzato. Quindi è possibile indirizzare un ingresso specifico di una tabella impiegando una specifica indicizzata per realizzare automaticamente un prelievo indiretto. In un processore potente dovrebbero essere disponibili gli indirizzamenti indicizzato ed indiretto.

Sfortunatamente, l'indirizzamento indiretto è disponibile solo su pochi computer. Esso può essere realizzato con certi limiti per l'8080 e lo Z80 per mezzo dei registri H ed L. Il Signetics 2650 fornisce il vero indirizzamento indiretto.

PROGRAMMAZIONE IN LINGUAGGIO ASSEMBLY

Ad eccezione dei casi più semplice, dove si programmerà direttamente in esadecimale (cioè in binario) la maggior parte della programmazione del microprocessore sarà fatta in assembler oppure in un linguaggio ad alto livello. Di seguito si descriveranno le caratteristiche di un assembler, i tipi di istruzioni disponibili, con i modi di indirizzamento appena descritti e che possono essere impiegati per modificare qualsiasi istruzione. Verranno quindi presentati dei programmi reali.

Si richiamano innanzi tutto le definizioni di un assembler. L'assembler è un programma che traduce automaticamente la rappresentazione simbolica delle istruzioni nella loro codifica binaria. La caratteristica di base di un assembler è la capacità di specificare le istruzioni in un formato mnemonico o simbolico. È possibile rappresentare, per esempio, l'accumulatore con A oppure ACC o per mezzo di qualsiasi altro simbolo. Ai registri si possono associare dei nomi. Il registro 0 potrebbe essere chiamato «Alfa», il registro 1, «Beta». Analogamente si possono assegnare

dei nomi ai dati da utilizzare. Esiste una caratteristica speciale all'interno dell'assembler che consente l'assegnazione di nomi a variabili e registri.

Un altro vantaggio dell'assembler è che, durante il processo di assembly, esso genererà diagnostici di errore ogni volta che si rivela una sintassi non corretta. Invece esso non rivelerà mai errori logici. Gli errori rivelati saranno sviste, errori di modo di indirizzamento, sulla struttura degli operandi, di doppia definizione di nomi simbolici. Un programma scritto in linguaggio assembly viene chiamato *programma sorgente.* Una volta tradotto dall'assembler in codice binario esso viene chiamato *codice oggetto.* La programmazione in linguaggio di livello assembly è caratterizzata dalla stessa efficienza di esecuzione della programmazione diretta in codice binario. Ogni istruzione di livello assembly sarà tradotta in un'istruzione binaria.

Un assembler fornisce due tipi diversi di istruzioni: *dichiarazioni* ed istruzioni *eseguibili.*

DICHIARAZIONI

Le dichiarazioni sono istruzioni speciali progettate per facilitare la scrittura del programma nel formato simbolico. Esse generalmente specificano l'equivalente binario dei simboli impiegati dal programma. Le dichiarazioni assegneranno un valore a costanti ed indirizzi. Per esempio la seguente dichiarazione:

ORG $1000

Specifica che il programma da considerare inizia all'indirizzo 1000 esadecimale (origine).

Ogni volta che viene impiegata una specifica di origine il programma risiederà ad un ben definito indirizzo della memoria. Questa è chiamata *programmazione assoluta.* Ogni volta che un programma può essere caricato in qualsiasi posizione della memoria è detto *programma ri-allocabile.* Per posizionarlo fisicamente in memoria sarà necessario convertire gli indirizzi simbolici impiegati nel programma in indirizzi fisici effettivi. Questo verrà realizzato da un caricatore. Nel caso dei microprocessori, al contrario rispetto ai minicomputer ed i grossi computer, quasi tutti i programmi sono caricati una volta per tutte in una memoria ROM che non può essere cambiata. Essi sono perciò caricati ad indirizzi assoluti. La possibilità di ricaricare un programma è essenziale durante la fase di sviluppo poichè non si conosce la lunghezza di ogni segmento di programma. Una volta che il programma è scritto completamente non esistono controindicazioni alla conversione in locazioni assolute. Si noti che il *debugger* normalmente fornito dal costruttore che sarà usato per verificare la corretta esecuzione del programma, deve consentire di specificare gli indirizzi *relativi all'inizio* del programma oppure si perderà il vantaggio dei programmi riallocabili. L'utente dovrà quindi calcolare manualmente l'indirizzo di ogni istruzione all'interno del proprio programma.

La specifica di un indirizzo verrà eseguita nel modo seguente:

ADR1 EQU $1000.

Questa specifica assegnerà l'indirizzo assoluto 1000 esadecimale al simbolo ADR1 impiegato nel programma.

La specifica dei *dati* comprende due parti: occorre definire il *formato* ed i *contenuti*. Per esempio:

DATA RMB 1

nel linguaggio assembly del 6800 riserverà un byte per i dati RMB. Analogamente:

TABLE RMB 50

allocherà 50 bytes nella struttura dati chiamata TABLE.

Naturalmente occorre poter assegnare dei valori ai dati:

VAL1 FCB $10
CAR1 FCC 'A'

queste istruzioni assegneranno valori effettivi ai dati. La prima istruzione assegna il valore 10 esadecimale al simbolo VAL1. La seconda istruzione assegna il codice di 8 bit corrispondente ad «A» alla variabile CAR1. Il vantaggio di quest'ultima è di convertire automaticamente il simbolo «A» nella codifica binaria simbolica corretta. Esistono altre caratteristiche per altri tipi di dati, quali ottali oppure decimali.

Infine ogni programma di livello assembly deve terminare con l'istruzione «END».

ISTRUZIONI ESEGUIBILI

Le istruzioni eseguibili sono le istruzioni considerate inizialmente, cioé quelle che il microprocessore eseguirà direttamente. Esse sono semplicemente rappresentate in modo simbolico.

Ogni istruzione di livello assembly è divisa in quattro campi:

1. Indirizzo Simbolico
2. Istruzione Mnemonica
3. Operando
4. Commento

Esempio:

BEGIN	LDAA	=	$ 4F	CARICA L'ACC A CON IL VALORE 4F
(indirizzo)	(codice operativo)		(operando)	(commento)

Il campo dell'*indirizzo* viene anche chiamato campo della *label*. Questo è molto

importante. Qualsiasi istruzione può ricevere una label simbolica invece di un indirizzo assoluto. Non tutte le istruzioni richiedono la label. Comunque questo è più conveniente per il riferimento delle istruzioni oppure per specificare le operazioni di salto. È possibile, per esempio, inserire facilmente un'istruzione addizionale all'interno del programma senza dover riscrivere tutte le altre operazioni di salto. Qualora gli indirizzi di diramazione fossero espressi in modo assoluto, sarebbe necessaria la loro modifica poichè tutte le istruzioni dovrebbero essere spostate in avanti nella memoria di un certo numero di parole. L'assembler invece, nel caso di labels, sostituirà automaticamente il corretto indirizzo numerico al posto di quello simbolico durante la fase assembly.

Inoltre, le labels forniscono una notevole convenienza nel riferimento ad istruzioni, o a gruppi di istruzioni, all'interno del programma. Naturalmente le labels, come la maggior parte degli altri simboli impiegati nell'assembler, sono limitate dalla lunghezza ed il numero dei simboli che esse possono impiegare per fornire una scrittura efficiente in assembly. Tipicamente esse sono limitate a sei caratteri.

Il campo mnemonico, chiamato anche codice operativo, deve sempre essere specificato. Esso rappresenta simbolicamente l'istruzione da eseguire. Nel caso dei microprocessori ad 8 bit, questa è una rappresentazione simbolica dei primi otto bit. Nell'esempio precedente il campo mnemonico è occupato da LDAA e questo significa che l'accumulatore deve essere caricato con i contenuti della locazione di memoria. Anche la locazione di memoria deve quindi essere specificata.

Il *campo operando* è quello più complesso. La sua sintassi varia a seconda dell'istruzione. Esso può contenere un indirizzo simbolico, un dato immediato, oppure anche una semplice espressione aritmetica. Inoltre dentro questo campo può essere impiegato un certo numero di simboli speciali. Il segno di uguale (=) è impiegato tipicamente per l'indirizzamento immediato. Il segno di dollaro ($) è impiegato per indicare un valore esadecimale. Le virgolette (' ') sono impiegate per denotare un simbolo alfanumerico.

Gli assemblers consentono l'impiego dei simboli più e meno (+, -) per specificare dei semplici calcoli di indirizzo come:

ADR + 2

Questa specifica fa riferimento all'indirizzo calcolato come il valore *ADR + 2*.

Nel caso dell'indirizzamento *indicizzato*, sarà specificato il registro indice, se questo non lo era già come parte del campo mnemonico. La specifica potrebbe essere:

ADR, X

Questo indica che l'indirizzo effettivo può essere ottenuto sommando il valore di ADR ai contenuti del registro indice X.

L'indirizzamento *indiretto* è normalmente specificato da un carattere speciale, quale un asterisco (*):

ADR*

indica che ADR deve essere interpretato come un indirizzo indiretto.

Infine il campo operando può contenere nomi simbolici di registri nel caso di istruzioni del tipo registro.

Il campo operando può essere vuoto se un'istruzione non richiede un operando: (normalmente) le istruzioni di 8 bit hanno un campo operando vuoto.

Il *campo del commento* è un campo opzionale disponibile per comodità del programmatore. Esso è fondamentale per la *leggibilità del programma*. L'utente scriverà in esso i commenti che possono chiarire l'esecuzione dell'istruzione. La maggior parte dei campi di commento sarà di lunghezza arbitraria, alcuni saranno limitati nel numero di caratteri che possono essere utilizzati. La disponibilità di questo campo è un vantaggio molto importante per realizzare programmi auto-documentati. Questa caratteristica non può esistere ai livelli esadecimale o binario. Naturalmente questo campo è completamente ignorato dall'assembler. Esso viene impiegato solo per il listing del programma.

Convenzioni

Le convenzioni differiscono da un assembler ad un altro a seconda della sintassi delle caratteristiche disponibili. Comunque tutti gli assemblers sono generalmente simili ed ogni utente che ne conosce uno è in grado di usare facilmente qualsiasi altro.

Una caratteristica addizionale che può, in particolare, essere fornita da un assembler è la macro. Una *macro* è semplicemente un nome assegnato ad un gruppo di istruzioni. È conveniente, nell'impiego di una rappresentazione simbolica che essa rappresenti diverse istruzioni piuttosto che una sola. Qualsiasi buon assembler è dotato della possibilità di macro *condizionale*, consentendo l'assembly condizionale di macros.

Tipi di istruzioni

Diversi sono i criteri che possono essere seguiti per la classificazione delle istruzioni all'interno di un processore. In questa sede non si considerano i modi di indirizzamento poichè verranno trattati come sotto-modi di ogni possibile istruzione. La classificazione più diretta è la distinzione in funzione del codice operativo, cioè in funzione del tipo di operazione da eseguire. Si distingueranno quindi sei categorie di istruzioni:

1. Istruzioni di trasferimento che coinvolgono dispositivi esterni (memoria o I/O).
2. Istruzioni di trasferimento tra registri (interni alla CPU).

3. Istruzioni aritmetiche.
4. Istruzioni logiche.
5. Istruzioni di scorrimento.
6. Istruzioni di controllo (diramazione e test).

Si noti che, mentre nel settore dei grossi computer è consuetudine distinguere le istruzioni di I/O dalle altre, questa distinzione è obsoleta nel settore dei microprocessori. Infatti in questo caso l'indirizzamento I/O viene normalmente eseguito esattamente allo stesso modo dell'indirizzamento della memoria.

Si analizzeranno ora questi sei tipi di istruzioni.

1. Istruzioni di Trasferimento da/a Dispositivi Esterni

Queste istruzioni sono responsabili del movimento dell'informazione tra un registro del microprocessore ed una specifica localizzazione esterna, come la memoria. Tale informazione può avere lunghezza 8 oppure 16 bit. Si considerano alcuni esempi.

Con il 6800:
LDAA ADR1
LDAB ADR2

Le istruzioni precedenti caricheranno i contenuti degli indirizzi ADR1 ed ADR2 rispettivamente nei registri A e B (A e B sono i due accumulatori interni del 6800). ADR1 può far riferimento ad una locazione di memoria oppure al registro di un chip I/O.

Le istruzioni inverse sono:
STAA ADMEM1
STAB ADMEM2

Le istruzioni precedenti memorizzeranno i contenuti di A e B rispettivamente agli indirizzi di memoria ADMEM1 ed ADMEM2.

Analogamente, con l'8080, le istruzioni
MOV r, M
MOV M, r

trasferiranno rispettivamente i contenuti di una locazione di memoria nel registro r ed i contenuti del registro r nella locazione di memoria. Nel caso di queste istruzioni specifiche, l'indirizzo di memoria è contenuto in H ed L. Esiste un'altra istruzione per specificare immediatamente il campo indirizzo. Comunque queste istruzioni si riferiscono esclusivamente all'accumulatore A.

Qualsiasi istruzione che fa riferimento ad un indirizzo di memoria può essere impiegata in uno qualsiasi dei modi di indirizzamento disponibili sul microprocessore. Per esempio, con il 6800, LDAA = 44 caricherà l'accumulatore con il valore immediato 44. 44 è un operando immediato che è caricato «così com'è» nell'accumu-

latore, piuttosto che interpretato come indirizzo di memoria.

Analogamente LDAA ADR, X caricherà nell'accumulatore i contenuti della locazione di memoria specificata dai contenuti del registro indice X a cui va sommato ADR.

Possono essere disponibili altre istruzioni speciali: lo Z80 è dotato di diverse istruzioni speciali come le istruzioni di trasferimento di blocchi di dati oppure la ricerca di una precisa struttura binaria all'interno di un blocco di memoria. L'8085 è inoltre dotato di ulteriori istruzioni simili.

2. Trasferimenti tra Registri

Queste istruzioni manipolano dati internamente alla CPU, senza riferimento a qualsiasi dispositivo esterno. Esse saranno eseguite molto più velocemente di qualsiasi istruzione relativa a dispositivi esterni.

Sul 6800 le istruzioni TAB e TBA trasferiscono dati tra i registri A e B. Sull'8080, MOV r 1, r 2 esiste qualsiasi trasferimento possibile tra due registri qualsiasi. Possono anche essere disponibili delle istruzioni speciali per facilitare dei tipi speciali di trasferimento: sullo Z80 l'istruzione EX DE, HL scambierà automaticamente i contenuti dei registri DE con i contenuti dei registri H ed L (16 bit).

3. Istruzioni Aritmetiche

A causa dell'area limitata disponibile sul chip per realizzare le funzioni di CPU, le istruzioni aritmetiche, nel caso dei microprocessori, sono generalmente limitate a somma, sottrazione, incremento e decremento più alcune istruzioni di supporto come DAA e manipolazione del riporto (carry). Idealmente le istruzioni aritmetiche potrebbero comprendere tutte le comuni istruzioni aritmetiche più, almeno, moltiplicazione e divisione. Tra le operazioni aritmetiche è naturalmente possibile discriminare ulteriormente in istruzioni che coinvolgono il trasferimento tra registri ed il trasferimento tra registri e dispositivi esterni.

Per esempio, nel caso dell'8080:

ADD r

specifica l'addizione del registro r con l'accumulatore.

ADD M

specifica l'addizione di una parola di memoria con l'accumulatore. Si assume che l'indirizzo di memoria si trovi nei registri H ed L.

In entrambi i casi il risultato è nell'accumulatore.

Con il 6800:

ADD A MEM, ADD B MEM

sommerà il valore contenuto nella locazione di memoria MEM con l'accumulatore

A oppure B.

L'8080, lo Z-80 e l'8085 forniscono tutti istruzioni speciali che operano su 16 bit, anche se si tratta di microprocessori a 8 bit. Questo originerà un significativo miglioramento di efficienza nella manipolazione di dati a 16 bit.

Per esempio, sullo Z-80: ADD HL, BC eseguirà la somma dei registri H ed L con i registri B e C (16 bit). La stessa istruzione esiste sull'8080 e sull'8085.

Tutti i microprocessori sono dotati di istruzioni per l'incremento ed il decremento dei contenuti di registri specifici. Questo è indispensabile per qualsiasi operazione di conteggio.

Con il 6800, l'istruzione INC MEM, A, oppure B e DEC, incrementerà, o decrementerà una locazione di memoria (oppure, i contenuti degli accumulatori A o B).

Sono disponibili delle istruzioni speciali INX e DEX per incrementare o decrementare il registro indice a 16 bit.

Analogamente l'8080, 8085 e Z-80 sono dotati di istruzioni INC e DEC che operano su registri a 16 bit.

Si è visto che, per facilitare le operazioni BCD, deve essere disponibile un'operazione di aggiustamento decimale. Essa virtualmente è disponibile su tutti i microprocessori.
Si nota subito l'assenza delle istruzioni di moltiplicazione e divisione. Le operazioni in virgola mobile non sono modi disponibili. Raramente esse sono disponibili anche sui minicomputer a causa della grande quantità di logica necessaria per la loro realizzazione. È auspicabile che i nuovi microprocessori, specialmente quelli a 16 bit, siano almeno dotati delle operazioni di moltiplicazione e divisione. Il TMS 9900 già le possiede.

A causa della non disponibilità delle istruzioni di moltiplicazione e divisione acquista un particolare valore la disponibilità di istruzioni di *passo di moltiplicazione* e di *passo di divisione*. Un passo di moltiplicazione è un ADD condizionale basato sul valore del bit carry (riporto). Inoltre deve essere disponibile la possibilità di *segno esteso*. Un segno esteso consiste nella duplicazione del bit segno durante uno scorrimento a destra. Questo è necessario nella manipolazione di numeri in complemento a 2. Sfortunatamente pochi microprocessori forniscono queste istruzioni.

4. Istruzioni logiche

Devono essere disponibili almeno quattro operazioni logiche: OR, AND, OR esclusivo (abbreviato XOR) e NOT (complemento). La tabella della verità per ciascuna di queste operazioni è riportata su Fig. 8-16. All'interno di una «tabella della verità» uno «0» rappresenta «falso» ed un «1» rappresenta «vero».

La funzione OR originerà un'uscita vera ogni volta che almeno un ingresso è vero.

La funzione XOR originerà un'uscita vera se e solo se uno degli ingressi è vero.
La funzione AND darà un'uscita vera solo se entrambi gli ingressi sono veri.
La funzione NOT opererà semplicemente il complemento dell'ingresso commutando uno «0» in un «1» e viceversa.

OR

A	B	A V B
0	0	0
0	1	1
1	0	1
1	1	1

(AND)

A	B	A B
0	0	0
0	1	0
1	0	0
1	1	1

XOR

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

NOT

A	Ā
0	1
1	0

Fig. 8-16: Tabelle logiche della verità.

Le istruzioni logiche saranno specificamente impiegate nel caso di operazioni d'ingresso - uscita o per forzare valori specifici nelle locazioni.

Un OR logico è impiegato per forzare un 1 in un'assegnata posizione di bit.

Esempio:
10000000 OR 00001100 = 10001100.

L'OR inclusivo ha forzato il valore 11 nelle posizioni di bit 2 e 3 della parola originaria. Esso garantisce che gli uni contenuti nella seconda parola saranno introdotti nella prima e viceversa.

L'istruzione XOR serve a verificare se due parole sono identiche. Ogni volta che i contenuti sono identici il risultato sarà 0. Se un bit qualsiasi fosse diverso, il risultato non sarà 0 ma conterrà un «1» in alcune posizioni di bit. Si può osservare dalla tabella della verità per lo XOR che il risultato è «0» solo se tutti i bit di A e B sono identici.

Questo può essere utilmente impiegato per verificare se il valore di un registro d'ingresso è cambiato. Se il risultato è 0 non si è verificato alcun cambiamento. Se il risultato non è 0 almeno una posizione di bit è stata modificata. Infatti le posizioni di bit che sono state modificate sono rivelati da uni nelle posizioni corrispondenti del risultato.

L'AND logico viene impiegato per mascherare una specifica posizione di bit all'interno del registro, cioè per forzare uno «0» su di essa. Come si può notare dalla tabella della verità di Fig. 8-16 si ha che per l'AND il risultato è sempre «0» se uno

qualsiasi degli operandi è «0». Il risultato è «1» solo se entrambi gli ingressi sono «1», cioè «non 0».

Esempio:

Solo il bit 5 è stato mascherato da uno 0 contenuto nel secondo byte. Esso è stato forzato al valore «0» nel risultato sulla destra dell'esempio.

Su tutti i microprocessori sono disponibili tutte queste quattro operazioni anche se i mnemonici possono essere diversi da un microprocessore all'altro. In particolare per l'operazione NOT viene spesso utilizzato COMP. Queste istruzioni logiche verranno impiegate di seguito nel capitolo dedicato alla logica programmata.

5. Operazioni di scorrimento

Le operazioni di scorrimento comprendono scorrimenti veri e propri, rotazioni ed operazioni simili destinate a variare l'ordine dei bit all'interno di un registro. Altre caratteristiche che possono essere disponibili sono per esempio lo «scambio di nibble», che scambia i quattro bit di sinistra con i quattro di destra. Le operazioni di scorrimento sono necessarie per le istruzioni di moltiplicazione e divisione e per verificare i contenuti di qualsiasi bit all'interno di un registro. Tipicamente esse opereranno solo sull'accumulatore (o sugli accumulatori). Uno scorrimento a sinistra di un bit è equivalente ad una moltiplicazione per 2, nel sistema binario. Questo verrà sfruttato nell'esempio seguente. Analogamente uno scorrimento a destra è equivalente ad una divisione per due. Si possono distinguere tre tipi di scorrimento:

1. Scorrimento logico, a destra oppure a sinistra.
2. Scorrimento aritmetico, a destra oppure a sinistra.
3. Rotazione a destra oppure a sinistra.

Scorrimento logico

Uno scorrimento logico è uno scorrimento «puro». Durante un'operazione di scorrimento a sinistra i contenuti del bit di estrema sinistra saranno espulsi dal registro e finiranno nel bit carry. Inversamente i contenuti del bit di estrema destra verranno azzerati.

Fig. 8-17: Scorrimento e rotazione.

Attenzione: nella maggior parte dei microprocessori è convenzione comune indicare con 0 il bit di estrema destra e 7 quello di estrema sinistra in modo che la posizione del bit corrisponde al peso binario. Alcuni costruttori impiegano la convenzione opposta.

Con il 6800 l'istruzione LSR origina uno scorrimento a sinistra nel bit carry ed il bit 7 sarà spostato di una posizione a destra. Uno spostamento a sinistra potrebbe essere eseguito esattamente allo stesso modo, salvo invertire la direzione. Questo è illustrato in Fig. 8 - 17.

Scorrimento Aritmetico

Gli scorrimenti aritmetici sono impiegati nella manipolazione dei numeri in complemento a 2. Nello scorrimento a destra di un numero in complemento a 2 è spesso necessario che il bit entrante sulla sinistra sia identico al bit segno, se il bit segno non è cambiato inavvertitamente. Se il numero è negativo il bit entrante sulla sinistra sarebbe 1. Questa caratteristica è chiamata *«segno esteso»*. Il 6800 fornisce questa istruzione denominata ASR. Il bit 7 viene semplicemente ripetuto ogni volta che i contenuti del registro vengono spostati a destra. Anche lo Z-80 è dotato di un'istruzione simile.

Rotazione

In una rotazione (a 9 bit) il bit uscente dal registro viene depositato nel carry. Il bit entrante nel registro viene prelevato dal valore precedente del carry. Fisicamente questo implica che il carry contiene le parti d'ingresso e d'uscita. Nella maggior parte dei microprocessori la rotazione è matematica ed a *9 bit*. Per esempio, sul 6800, le istruzioni ROL e ROR sono rotazioni a 9 bit. Esse comprendono il bit C.

L'8080 e lo Z-80, oppure l'8085 sono dotati di un'ulteriore istruzione di addizione che ruota solo l'accumulatore, cioè solo 8 bit: RLC, RRC. Inoltre il bit C viene posto uguale al bit eliminato. Comunque l'8080 non è dotato di uno scorrimento «vero».

6. *Istruzioni di Controllo*

Le istruzioni di controllo servono a cambiare l'ordine in cui viene eseguito un programma, sia incondizionatamente oppure in dipendenza del valore degli indicatori di stato. Le istruzioni di controllo forniscono le cosiddette caratteristiche «intelligenti». Queste istruzioni differenziano un *computer* da un semplice *calcolatore*. Esse danno la possibilità al computer di realizzare decisioni diverse (programmi diversi) in dipendenza del valore di parametri misurati. Si possono distinguere tre tipi di istruzioni di controllo:

1. *Diramazioni Incondizionate*

Una diramazione incondizionata, o salto, è semplicemente un'istruzione GO TO

(vai a) addr. Essa forza un salto ad un locazione di memoria specifica. Nel caso dell'8080 esiste l'istruzione equivalente: JMP addr.

L'istruzione eseguita dopo JMP sarà quella all'indirizzo addr. Concettualmente, e spesso praticamente, l'indirizzo letterale addr sarà forzato nel contatore di programma. Questo fa prelevare automaticamente l'istruzione successiva all'indirizzo addr.

Nel caso del 6800 vengono fornite due istruzioni di diramazione: BRA e JMP. JMP è l'istruzione di salto generale mentre BRA è impiegata per eseguire un salto a breve distanza in un campo di 8 bit. D'altra parte l'8080 è dotato di un ulteriore istruzione di salto breve, l'istruzione RST che specifica, all'interno di un codice di 8 bit un indirizzo di salto di 3 bit, in multipli di 8 parole, nell'ambito delle prime 256 parole di memoria. Questo viene impiegato essenzialmente nel caso di interrupts.

2. *Salti Condizionati*

I salti condizionati vengono impiegati ogni volta che deve essere verificata una condizione. Normalmente si verifica la condizione di uno dei bit del registro di stato. Se il bit è 1 (oppure se la condizione è soddisfacente) si verifica la diramazione. Se la condizione non è soddisfatta, oppure se il bit è 0, non si verifica la diramazione ed il programma continua normalmente in sequenza.

I bit di stato, all'interno del registro di stato, vengono normalmente posizionati in modo automatico dalle istruzioni. Ogni costruttore fornisce una lista dei flags di stato che saranno influenzati da ciascuna istruzione. Talvolta è possibile posizionare bit specifici impiegando opportune istruzioni. Sul 6800 è disponibile un'istruzione speciale TST. Essa viene realizzata internamente mediante sottrazione esplicita del valore 0.

Normalmente ogni bit dei flags di stato può essere verificato singolarmente, indipendentemente per il valore 0 o per il valore 1 ed originare una diramazione. Le istruzioni di diramazione corrispondenti avranno nomi diversi a seconda del costruttore. Talvolta sono disponibili alcune istruzioni sofisticate, come la verifica di combinazioni di bit oppure combinazioni di condizioni, come un bit deve essere «maggiore o uguale a» oppure «minore o uguale a».

Nel caso del 6800, BEQ originerà una diramazione se Z=1, invece BPL se N=0.

Ogni istruzione di test che compare in un diagramma di flusso, sarà realizzata da una o più diramazioni convenzionali, nel programma. Esse rappresentano l'equivalente dell'operazione «IF ... THEN ... ELSE» («SE... ALLORA ... OPPURE ...») impiegata in un linguaggio ad alto livello.

Naturalmente esse realizzano solo scelte binarie cioé decisioni «sì oppure no». Ogni volta che è necessaria una decisione più complessa, e nel caso della programmazione in linguaggio assembly, si dovrà procedere ad una conversione in una suc-

cessione di test binari.

Un caso speciale di diramazione multipla è una diramazione ad 8 vie che dipende dal valore di un bit all'interno dell'accumulatore oppure di un registro. Sfortunatamente tale caratteristica non è disponibile nella maggior parte dei microprocessori. Sta diventando disponibile sui microprocessori più recenti. Essa è particolarmente efficiente per prendere una decisione su otto possibili.

ISTRUZIONI SPECIALI

Le istruzioni speciali, che non possono essere classificate nelle due categorie precedenti, comprendono:

- Varie istruzioni di sospensione ed arresto
- Possibilità di subroutines
- Direzione interrupt
- Direzione di altre caratteristiche che possono essere disponibili sul chip del microprocessore come un temporizzatore programmabile.
- Ogni volta che sono disponibili istruzioni specifiche d'ingresso ed uscita è possibile classificarle come speciali. Infine le istruzioni di manipolazione dello stack sono in realtà un caso speciale dell'indirizzamento implicito. Comunque esse cambiano automaticamente i contenuti del puntatore dello stack e come tali vanno considerate istruzioni speciali. Esse verranno descritte di seguito.

Le istruzioni speciali miste suddette sono generalmente autoesplicative e non richiedono nessuna spiegazione speciale, tranne le subroutines e la manipolazione dello stack. Verranno ora spiegati questi due gruppi di istruzioni:

SUBROUTINES

Un programma utente è formato da una sequenza di istruzioni che verranno depositate, per l'esecuzione, all'interno della memoria del microprocessore.

Una *subroutine* è un gruppo di istruzioni (un sottoprogramma) all'interno del programma principale, aventi un nome specifico e delimitate da due istruzioni specializzate: SUB all'inizio e RETURN alla fine.

La Fig. 8-18 illustra l'impiego di subroutines. Normalmente le subroutines vengono composte dall'assembler alla fine del programma principale. Comunque esse potrebbero risiedere ovunque in memoria, analogamente ad un qualsiasi segmento di programma.

Il meccanismo di impiego delle subroutines è il seguente: nel corso del programma principale (parte sinistra della Fig. 8-18) si incontrerà un'istruzione speciale: CALL SUB 1

L'effetto di una chiamata di subroutine originerà un salto all'indirizzo di partenza della subroutine SUB 1. Il controllo viene «trasferito» alla subroutine. Questo

non è però il solo effetto dell'istruzione CALL. Per il momento si tralasci questo punto. Verrà quindi eseguita la subroutine SUB 1. Si ignori per il momento la subroutine SUB 2 che appare sulla destra dell'illustrazione. Si assuma inizialmente che la chiamata di subroutine SUB 2 non sia presente all'interno di SUB 1. Verrà quindi eseguita SUB 1 finché non si incontra l'istruzione RETURN. L'effetto dell'istruzione RETURN sarà il «ritorno al programma principale». L'istruzione successivamente eseguita è quella che segue CALL SUB 1 nel programma principale.

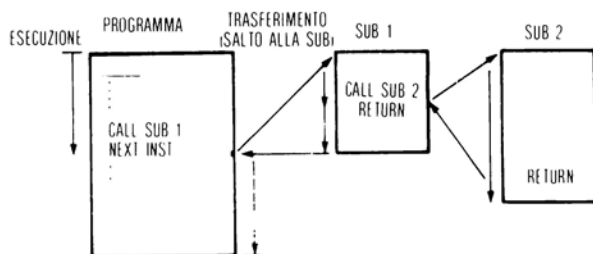



Fig. 8-18: Programma e subroutine.

Concettualmente l'effetto è analogo all'inserzione di SUB 1 al posto dell'istruzione CALL. Per il ritorno al programma principale occorre conservare l'indirizzo di ritorno. Nell'istante di chiamata della subroutine, il contatore di programma è già stato incrementato e contiene l'indirizzo dell'istruzione successivamente eseguibile, cioè l'indirizzo di «NEXT INST». Dopo aver incontrato l'istruzione CALL è necessario conservare i contenuti di PC. Questi verranno conservati nello stack e questo è l'altro effetto, precedentemente ricordato, dell'istruzione CALL.

Si può eseguire un numero qualsiasi di volte la chiamata successiva di subroutine. Si faccia ora riferimento all'illustrazione effettiva di Fig. 8-18. Nel corso dell'esecuzione di SUB 1 si incontra un'altra chiamata CALL SUB 2.

Questo origina la chiamata alla subroutine SUB 2 sulla destra della Fig. 8-18. Invece di continuare l'esecuzione all'interno di SUB 1, il controllo viene trasferito alla SUB 2 e l'istruzione successivamente eseguita è la prima istruzione eseguibile di SUB 2.

Si supponga che SUB 2 non contenga ulteriori chiamate di subroutine. Quindi SUB 2 verrà normalmente eseguita fino ad incontrare l'istruzione speciale RETURN. Il RETURN originerà un ritorno all'istruzione successiva all'interno di SUB 1, seguente la chiamata di SUB 2. L'esecuzione proseguirà quindi all'interno di SUB 1 fino ad incontrare RETURN. Questa infine originerà un ritorno al programma chiamante ed all'esecuzione di NEXT INST.

Normalmente nulla vieta ad una subroutine di chiamare se stessa. Si dice allora che si considera una chiamata di subroutine *recursiva*. In questo caso si assume che venga usato un contatore o qualche altro meccanismo all'interno del programma in modo, eventualmente, da terminare l'esecuzione.

Il meccanismo della subroutine

Il meccanismo di realizzazione della «chiamata della subroutine» è abbastanza semplice: quando si incontra un'istruzione CALL, il registro PC viene conservato nello stack. Ogni istruzione CALL conserva PC e quindi carica in PC stesso l'indirizzo di inizio della subroutine.

L'istruzione RETURN ripristina PC. Essa preleva semplicemente le due parole alla sommità dello stack posizionandole in PC. Quindi l'esecuzione procede normalmente.

Ogni «livello» di subroutine origina un nuovo ingresso nello stack. Inoltre una subroutine richiederà normalmente dei «registri di lavoro» per la sua esecuzione. Il programmatore deve quindi provvedere a caricare nello stack i contenuti dei registri necessari all'interno della subroutine. Questi registri devono essere ripristinati dal programmatore al ritorno dalla subroutine. Un esempio di questa procedura è riportato più avanti nel caso: «la manipolazione di interrupt sull'8080».

Ogni volta che, all'interno di una subroutine, si incontra una nuova chiamata di subroutine, si dice che si deve creare un nuovo «livello di subroutine», cioè un nuovo ingresso allo stack o «livello». Per ogni livello sono necessarie almeno due parole (per la memorizzazione del PC). In generale sono richieste più parole per conservare i registri che la subroutine non dovrebbe cancellare. È responsabilità del programmatore riservare nella memoria un'area di stack sufficiente.

Il vantaggio di una subroutine è che essa può essere scritta solo una volta ed eseguita un numero qualsiasi di volte da programmi diversi. Essa quindi rappresenta un notevole *risparmio di memoria*. Inoltre essa contribuisce in modo sostanziale alla comprensione di un programma.

Lo svantaggio di una subroutine risiede nelle *istruzioni aggiuntive* per la sua inserzione nel programma. Infatti occorre eseguire le istruzioni CALL e RETURN. Se una subroutine è molto breve e la sua esecuzione è fondamentale, come nel caso delle routine di moltiplicazione e divisione, è preferibile ripetere semplicemente il gruppo di due o tre istruzioni alcune volte piuttosto che impiegare una subroutine. In questo caso si cerca quindi di aumentare l'efficienza a scapito di un maggiore impiego di memoria.

Un altro vantaggio derivante dall'impiego di subroutine è che, una volta scritte, esse possono essere condivise da utenti diversi. È possibile istituire una biblioteca di subroutine. Ogni subroutine può essere collaudata individualmente e questo è un vantaggio significativo.

Attenzione: una subroutine per operare utilizzerà dei registri. Essa potrebbe usare alcuni registri già impiegati dal programma principale. Ogni utente dovrebbe prendere visione di quali registri sono impiegati da una certa subroutine e regolarsi conseguentemente.

Una subroutine potrebbe richiedere l'impiego di registri non disponibili, in quanto già utilizzati dal programma principale. In questo caso il programmatore dovrebbe preservare tali registri nello stack oppure nella memoria e ripristinarli alla fine della subroutine.

La fase di ritorno non è una semplice diramazione. Occorre ripristinare il registro di stato ed il contatore di programma. Questo implica un'operazione di prelievo dallo stack durante la quale il valore del contatore di programma, preservato sulla sommità dello stack, verrà ritrasferito nello stesso registro, dando origine al trasferimento automatico all'istruzione successiva quella della chiamata di subroutine. Questo meccanismo può essere ripetuto finché é disponibile dello spazio nello stack.

A titolo di esempio si esamini il funzionamento della subroutine nel caso dell'8080. Una chiamata di subroutine viene realizzata da: CALL SUBAD, dove SUBAD è l'indirizzo di inizio della subroutine. L'esecuzione della chiamata provvederà automaticamente a caricare sulla sommità dello stack il valore corrente di PC. La prima parola disponibile sulla sommità dello stack è puntata dal registro SP. Tale registro viene quindi decrementato di due automaticamente e l'esecuzione viene trasferita all'indirizzo SUBAD. Il puntatore dello stack deve essere decrementato di *due* locazioni poichè si sta caricando nello stack i contenuti di un registro a 16 bit (PC). Nel caso dell'8080, per convenzione, gli indirizzi *decrescono* all'aumentare dello stack. Una *crescita* dello stack è caratterizzata da un *decremento* del puntatore dello stack.

In modo inverso si realizza il ritorno mediante RET.

L'esecuzione di RET provoca un prelievo automatico delle due parole alla sommità dello stack ed il registro SP viene contemporaneamente incrementato di due.

Le istruzioni equivalenti nel caso del 6800 sono: JSR (Jump to Subroutine: salta alla subroutine) ed RTS (Return from Subroutine: ritorna dalla subroutine).

ISTRUZIONI PER LO STACK

Il ruolo dello stack nel caso degli interrupts è già stato considerato al Capitolo 3. Nel caso delle subroutines lo stack viene impiegato per MEMORIZZARE il valore del registro. Il numero massimo di chiamate che possono essere realizzate è determinato dalla lunghezza dello stack.

Lo stack può anche essere vantaggiosamente impiegato durante l'elaborazione ordinaria. In particolare lo stack viene impiegato per trasferire velocemente alla

memoria dei blocchi di dati. Una singola istruzione PUSH (caricamento nello stack) originerà un trasferimento di dati ad un indirizzo di memoria. In questo modo non sono necessarie istruzioni addizionali per il posizionamento e l'incremento degli indirizzi di memoria.

Lo stack viene anche impiegato per la conservazione dei registri richiesti all'interno di una certa subroutine. In questo caso il programmatore può quindi trasferire nello stack i contenuti dei registri richiesti e riutilizzarli in tempi successivi. Questo è quanto avviene nella manipolazione di interrupts.

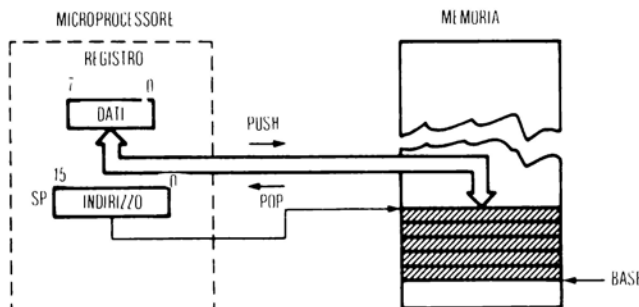



Fig. 8-19: Introduzione (PUSH) e prelievo (POP) dallo stack.

Le istruzioni relative allo stack per il 6800 sono: PSHA e PSHB. Queste istruzioni caricano rispettivamente, i contenuti dell'accumulatore A e B alla sommità dello stack. Viceversa PULA oppure PULB rimuoverà la parola alla sommità dello stack trasferendola in A oppure B. Il registro SP verrà automaticamente incrementato o decrementato.

Analogamente l'8080 e lo Z80 sono in grado di caricare o prelevare elementi dallo stack. Inoltre essi sono dotati di un sistema di incremento e decremento che può funzionare con qualsiasi coppia di registri. In questo modo una coppia di registri qualsiasi può essere impiegata come pseudo - puntatore dello stack.

Ogni volta che è disponibile l'auto-indicizzazione, qualsiasi coppia di registri può essere impiegata come un potenziale puntatore dello stack, anche se non sono disponibili le caratteristiche esplicite di caricamento e prelievo dallo stack. Per esempio il microprocessore SCMP della National è dotato di tre registri puntatori P1, P2, P3 che possono essere impiegati come puntatori dello stack impiegando l'auto-indicizzazione.

L'*auto - indicizzazione* consiste di un indirizzamento indicizzato con un incremento/decremento automatico del puntatore associato.

Verranno presentati di seguito diversi programmi eseguibili sull'8080 e sul 6800. Alla fine del libro sono riportati i rispettivi sets di istruzioni. Si considererà inizialmente un esempio aritmetico illustrante la manipolazione dei registri interni ed il trasferimento dati all'interno della MPU. Sucessivamente verrà presentata la realizzazione ingresso-uscita di alcuni esempi tra cui un programma effettivo per la generazione di musica mediante microprocessore. Questi esempi serviranno ad illustrare i concetti fin qui considerati e ad evidenziare alcuni problemi dello sviluppo di programmi effettivi. Questi sono soltanto alcuni esempi ma si raccomanda una pratica molto più estesa.

ESEMPIO: Manipolazione Interrupt con l'8080

Ogni operazione di Manipolazione Interrupt consiste nella memorizzazione iniziale dei registri e nel loro ripristino finale. Per memorizzare i registri si impiega:

```
PUSH PSW
PUSH H
PUSH D
PUSH B
```

Si noti che PUSH memorizza 2 registri: PUSH PSW memorizza PSW (flags) ed A. PUSH H memorizza H ed L, ecc.

Per ripristinare i registri, alla fine, si usa:

```
POP B
POP D
POP H
POP PSW
```

Naturalmente i registri vanno ripristinati nello stesso ordine della memorizzazione.

DOMANDA: *Perchè PC non viene memorizzato nello stack? Ed SP?*

UN PROGRAMMA ARITMETICO: LA MOLTIPLICAZIONE

Viene eseguita una moltiplicazione fra interi di 8 bit sull'Intel 8080. Inizialmente occorre sviluppare un algoritmo di moltiplicazione. A tale scopo si consideri la normale tecnica di moltiplicazione. Si moltiplichi 13 × 12:

```
     13   moltiplicando
x    12   moltiplicatore
  -----
     26
    13
  -----
    156        risultato
```

L'algoritmo della moltiplicazione è il seguente: si considera il moltiplicatore. Se

la sua cifra di estrema destra (LSB) non è zero la si moltiplica per il moltiplicando ed il risultato va scritto in modo da essere sommato al prodotto parziale. Se questa cifra è 0 si passa alla cifra successiva spostando il prodotto parziale a sinistra di una posizione.

In ogni caso, se la cifra del moltiplicatore è 0 o no, il numero successivo da sommare viene spostato a sinistra. Se la cifra è 0 non si dovrà sommare nulla. Se la cifra non è 0 avverrà la somma. Con questa tecnica si risolve quindi il problema della moltiplicazione sostituendola con una sequenza di addizioni. Questo è l'algoritmo più comune. In questo caso uno spostamento a sinistra è equivalente ad una moltiplicazione per 10.

Nel caso di numeri binari l'algoritmo che si utilizza è lo stesso. Si consideri il caso della moltiplicazione 3 per 5:

```
Moltiplicando                  011    (3)
Moltiplicatore               x 101    (5)
                             -----
Verifica di «1» su 101:        011
Verifica di «0» su 101:       000
Verifica di «1» su 101:      011
                             -----
Risultato                    01111   (15)
```

Nel sistema binario il bit più a destra del moltiplicatore, LSB, è il primo ad essere verificato. Se esso è uguale ad 1, il moltiplicando viene sommato, se uguale a 0 tale somma non viene eseguita. In ogni caso, la successiva addizione del moltiplicando verrà eseguita con un movimento a sinistra di 1 bit.

Questo algoritmo viene leggermente modificato poichè in un processore è più conveniente accumulare una somma parziale di una memorizzazione di numeri da sommare alla fine (manualmente è più comodo il contrario) e così di volta in volta si somma il moltiplicando al *prodotto parziale*. In questo modo si deve memorizzare solo il prodotto parziale. Per la somma del moltiplicando al prodotto parziale esistono due modi equivalenti: fare scorrere a sinistra il moltiplicando oppure a destra il prodotto parziale. I due risultati sono identici.

In questo modo la moltiplicazione binaria viene normalmente eseguita mediante una sequenza di addizioni e scorrimenti. La programmazione standard impiega un artificio per migliorare l'efficienza e diminuire il tempo di calcolo. Infatti non è strettamente necessario un test diretto del bit meno significativo. Per il test di questo bit sarà necessario far scorrere a destra, in sequenza, tutti i bit del moltiplicatore. Quindi le posizioni dei bit saranno disponibili sulla sinistra del registro contenente il moltiplicatore.

Inversamente si creerà un prodotto parziale ad ogni passo. Inoltre ad ogni passo il prodotto parziale aumenterà di un bit. Inizialmente esso sarà contenuto in un registro di 8 bit, ecc. Invece di impiegare un registro addizionale per memorizzare il

prodotto parziale, si userà il registro moltiplicatore per spostare in esso il prodotto parziale che viene fatto scorrere a destra.

In ogni istante dell'esecuzione della moltiplicazione, il registro inizialmente riservato al moltiplicatore conterrà, sulla destra, una parte del moltiplicatore e, sulla sinistra, la parte meno significativa del prodotto parziale. Questo richiederà l'impiego di un contatore di bit. In questo caso si impiegherà il registro E dell'8080 per la memorizzazione del bit di conteggio. Esso verrà inizializzato al valore 9, per essere decrementato a 0 quando la moltiplicazione è completa. Inizialmente i registri C e D conterranno il moltiplicatore ed il moltiplicando. Alla fine della moltiplicazione il risultato sarà contenuto in C e D.

Fig. 8-20: Moltiplicazione 8 bit per 8 bit (8080).

La Fig. 8-20 rappresenta il programma che traduce questo algoritmo nel programma in linguaggio assembly per l'8080.

MVI, B,0

Quest'istruzione carica, in modo immediato, l valore 0 in B. Occorre infatti originare un valore iniziale per il prodotto parziale, cioé inizializzarlo al valore 0.

Questo è un esempio di istruzione *immediata* dove la parola da caricare nel registro compare al secondo byte dell'istruzione (in questo caso 00000000).

MVI E,9

In modo analogo viene caricato il valore 9 nel contatore E. Si verifichi, alla fine

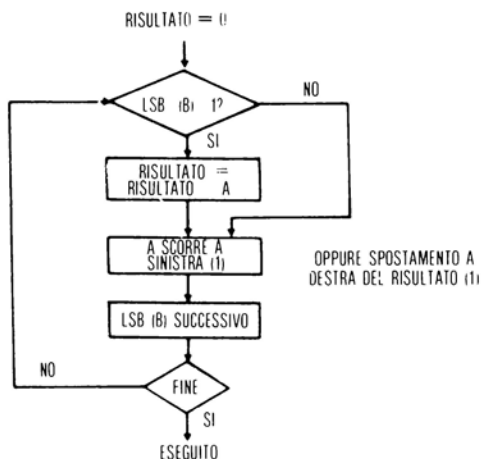



Fig. 8-21: Algoritmo di base della moltiplicazione.

dell'esempio, che il valore 9 sia corretto.

MOV A,C

Questa istruzione trasferisce i contenuti del registro C nell'accumulatore. Esiste questo problema: si vuole eseguire un test sul bit meno significativo del moltiplicatore, che si suppone contenuto in C. Sfortunatamente l'8080 non consente il test di un bit di un registro generico, ma soltanto dell'accumulatore. È quindi necessario, per il test di questo bit, ricopiare nell'accumulatore i contenuti del registro C, eseguire lo scorrimento e ricaricare i contenuti dell'accumulatore nel registro C. Tutto questo viene eseguito dalla terza, quarta e quinta istruzione:

RAR

Questa istruzione fa ruotare a destra l'accumulatore. Questo appare nella Fig. 8-21 in alto. I contenuti dell'accumulatore (uguali ai contenuti precedenti del registro C, cioè al moltiplicatore) vengono ruotati a destra di una posizione. Il bit più a destra del moltiplicatore si porta nel bit carry C dove avviene il test. È importante notare cosa accade al bit dell'accumulatore proveniente dal carry. Si vedrà così che questo è il meccanismo impiegato per trasmettere il prodotto parziale alla parte sinistra del registro C, in modo automatico.

Eseguita la rotazione i contenuti dell'accumulatore possono essere ricopiati in C. I contenuti del bit carry C verranno cambiati da un'operazione di scorrimento:

MOV C,A
DCR E

In questo modo si decrementa di un'unità il contatore di bit. A questo punto esso contiene il valore 8. Ogni volta che viene eliminato un bit del moltiplicatore, tale contatore viene decrementato di 1.

JZ DONE

JZ significa salta se Zero (Jump on Zero). Il processo è terminato se il contatore è stato decrementato al valore 0 ed è necessario saltare alla label «DONE» che appare nella Fig. 8-21 in basso. Il programma passerà quindi ad eseguire l'istruzione avente tale label. È interessante notare che il decremento ed il test del contatore viene eseguito prima di un'eventuale addizione al prodotto parziale. Occorrerà eseguire il test e sommare per otto volte in quanto il moltiplicatore contiene otto bit. La prima volta che si percorre il programma il contatore ha il valore 8, l'ultima volta il valore 1. Questo perchè si è partiti con il contatore di bit uguale a 9. Questo punto verrà verificato nuovamente nel corso dell'esercizio. Si consideri nuovamente il programma:

MOV A,B

Ora B contiene il prodotto parziale. Inizialmente esso è stato inizializzato al valore 0 ma poi i suoi contenuti saranno diversi da 0, avendogli sommato il moltiplicando. Questo può essere eseguito solo se il prodotto parziale è contenuto nell'accumulatore. È quindi necessario trasferire nell'accumulatore i contenuti di un registro per eseguire l'istruzione successiva: B viene trasferito in A.

JNC MULT1

Nessuna delle istruzioni precedenti ha modificato il valore del bit C. JNC significa «Jump if No Carry» (Salta se no Carry). Se il Carry è 0 non si deve sommare il moltiplicando al prodotto parziale e si salta direttamente alla label MULT 1 che è la seconda istruzione successiva. In ogni caso, se il carry è uguale ad 1, cioè se lo è il bit più significativo del moltiplicatore, (memorizzato in C) è necessario sommare il moltiplicando e quindi si esegue l'istruzione sequenzialmente successiva:

ADD D

Questa istruzione esegue semplicemente la somma dei contenuti del registro D, cioè del moltiplicando, all'accumulatore, contenente il prodotto parziale. L'addizione modificherà il valore del bit C, supponendo di moltiplicare numeri positivi, originando un carry uguale a 0.

RAR

Mediante questa istruzione i contenuti dell'accumulatore sono ruotati a destra di una posizione. Il bit più significativo del prodotto parziale viene depositato come nuovo valore del carry. La Fig. 8-21 in alto mostra questa rotazione eseguita sul

registro B, in quanto è necessario copiare semplicemente i contenuti dell'accumulatore nel registro B:

MOV B,A

Questa istruzione ritrasferisce i contenuti dell'accumulatore nel registro B.

Il ciclo si è così completato e basta ritornare alla sua prima istruzione:

JMP - MULT 0

Si tratta di un salto incondizionale che forza un ritorno all'indirizzo MULT 0, cioè alla quarta riga del programma.

Nuovamente si ha una rotazione a destra di C per il test e così via.

Per verificare che questo programma esegua effettivamente una moltiplicazione di due numeri di 8 bit si consiglia la *verifica manuale* del programma per mezzo della Tabella 8-22. La tabella va riempita procedendo in questo modo: sulla sinistra si riporta l'istruzione da eseguire. Successivamente si scrive in ciascuna delle sei colonne il valore del bit C e degli altri cinque registri nell'istante in cui l'istruzione è appena stata eseguita. Questo va ripetuto per ogni istruzione. Si parta dalla moltiplicazione di 5 (in C) per 3 (in D). Il risultato dovrebbe essere 15 in binario. Occorre controllare, con molta attenzione, cosa succede al bit C dopo l'esecuzione di ciascuna istruzione. Alcune istruzioni modificheranno il bit , altre lo lasceranno intatto. Per convenienza questa tabella può essere estesa su un foglio a parte.

	C	A	B	C	D	E
MULT		7 0	7 0	7 0	7 0	7 0
		00000000	00000000	00000101	00000011	— — —

Fig. 8-22: Esercizio di programmazione.

SOMMARIO

Questo programma di moltiplicazione illustra l'impiego di un tipo importante di istruzioni. Esso mostra come viene trasferita l'informazione tra i vari registri della macchina. Si sono rilevati anche alcuni inconvenienti di questo microprocessore: poicnè è impossibile il test diretto di un bit di un registro qualsiasi, i registri devono essere ricopiati nell'accumulatore prima del test dei loro contenuti. Risulta quindi indispensabile il trasferimento tra i registri e l'accumulatore, riducendo l'efficienza del programma. Praticamente tutti i microprocessori, attualmente disponibili sul mercato, hanno questo inconveniente.

Dopo aver mostrato come un algoritmo viene eseguito all'interno della MPU, si considererà la connessione del microprocessore al mondo esterno e l'esecuzione effettiva di operazioni d'ingresso-uscita.

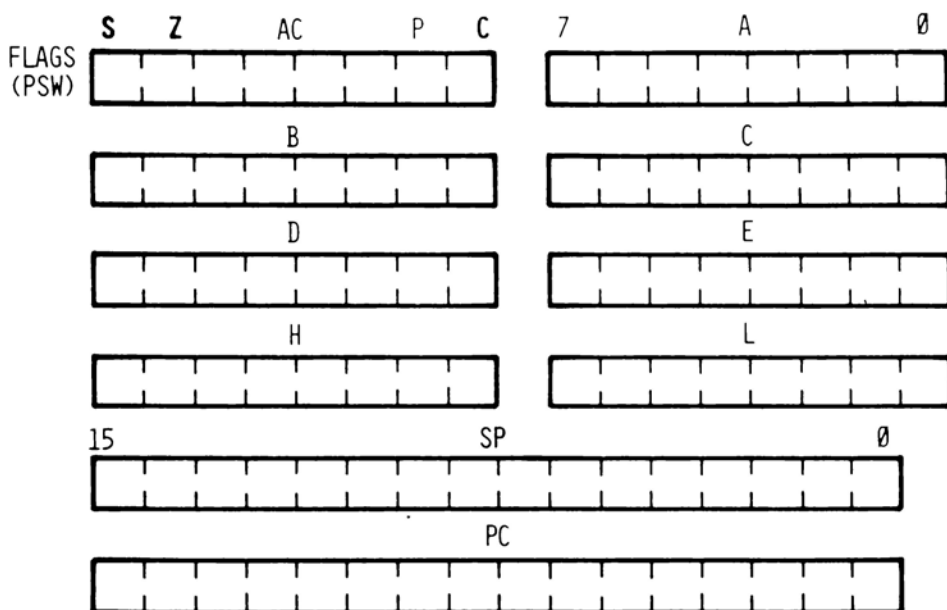



Fig. 8-23: Foglio di lavoro dei registri dell'8080.

SIMULAZIONE DELLA LOGICA DIGITALE MEDIANTE PROGRAMMA

Ogni microprocessore è dotato di un insieme di istruzioni logiche quali AND oppure NOT. Per questo esso è in grado di realizzare l'equivalente di una funzione logica qualsiasi, realizzata normalmente con queste porte. Quindi la logica combina-

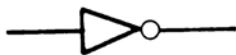
toria, o sequenziale, può essere sostituita da un programma equivalente. Si fa notare che, benchè il concetto sia corretto, questo può trarre in inganno. Infatti le porte possono essere sostituite da programmi ad un livello uno ad uno. Questo potrebbe essere il tipo peggiore di programmazione, caratterizzato da una grande inefficienza. È imperativo che l'approccio alla programmazione venga realizzato in un modo completamente diverso. La programmazione consente la sostituzione completa dei moduli funzionali con una soluzione programmata. Le porte logiche potrebbero essere un punto di partenza molto semplice. Comunque questo esempio non ha un valore educativo. Per questo motivo di seguito si mostreranno solo alcuni esempi della sostituzione di funzioni logiche con un programma equivalente.

Si può notare che una realizzazione della logica mediante programma viene realizzato proprio nel caso di un'unità di controllo implementato da microprogramma. Il microprogramma rappresenta una sostituzione diretta della realizzazione con logica casuale (random). In ogni caso non si devono necessariamente realizzare le stesse sequenze o le stesse tecniche. Esiste un'equivalenza a livello *funzionale*. Diversi sono i vantaggi della realizzazione programmata delle funzioni logiche: basso costo, velocità di realizzazione, facilità di debugging e flessibilità. Vengono eliminati completamente gli accessori hardware come buffers e drivers. Inoltre, sempre nel caso della realizzazione con programma, sono eliminati i limiti di fan-in e fan-out ed altri limiti elettrici.

Per mostrare questa tecnica verranno presentati quattro esempi: un invertitore, le porte AND/OR, un flip-flop ed un ritardo.

INVERTITORE REALIZZATO MEDIANTE PROGRAMMA

Un invertitore realizza la funzione NOT. Un segnale in ingresso a livello logico alto è convertito in un'uscita bassa e viceversa.

La rappresentazione simbolica del circuito è la seguente:

Un segnale alto verrà codificato come un «1» logico. Un segnale basso verrà considerato come «0» logico. In un programma la funzione di inversione sarà realizzata invertendo il valore di un bit. Questo viene realizzato molto semplicemente. Questo esempio sarà realizzato nel linguaggio assembly del 6800. Alla fine del libro è riportato un elenco delle istruzioni del 6800 con la loro funzione.

```
LDA A    SEGNALE    CARICA IL VALORE DEL SEGNALE
COM A               COMPLEMENTA A
STA A    SEGNALE    MEMORIZZA IL VALORE COMPLEMENTATO
                    NELLA VARIABILE SEGNALE
```

La variabile simbolica «segnale» del programma precedente fa riferimento ad una locazione di memoria. Si può anche impiegare un registro del 6800. La prima istruzione del programma caricherà nell'accumulatore il valore della cella di memoria all'indirizzo SEGNALE. Si assume che il valore del segnale d'ingresso sia depositato nella cella denominata «SEGNALE». In un sistema 6800 standard questo potrebbe essere realizzato da un registro d'ingresso PIA. Il segnale di indirizzo potrebbe far riferimento ad una porta del PIA.

Si è ottenuto un programma formato da 3 istruzioni che complementa il valore ad 8 bit del segnale. Questo può non essere quanto si voleva ottenere. Se si vuole complementare solo 1 bit, occorre impiegare una *maschera* per separare la posizione scelta. L'impiego di una maschera verrà esposto a proposito dell'«AND logico».

Si assuma che il segnale da invertire sia connesso al bit 2 di una porta del PIA. L'inversione del segnale effettivo sarà eseguita dalla seguente sequenza:

```
LDA      SEGNALE
TAB                      CARICA B IN A
COMA                     COMPLEMENTA A
ANDA     #02             CARICA IL BIT 2 IN A
ANDB     #$FD            AZZERA IL BIT 2 IN B
ABA                      SOMMA A + B
STAA     SEGNALE
```

In questo programma il simbolo «#» indica indirizzamento immediato. Invece «$» indica un valore esadecimale.

L'operazione «AND» eseguita sulla quarta riga di questo programma maschera tutti i bit tranne il bit 2 che può quindi essere sommato nell'accumulatore B dove tutti gli altri bit sono stati posti a 0.

È necessario un miglioramento al programma di campionamento per un suo impiego efficiente. In realtà, una porta singola su un PIA non verrà impiegata contemporaneamente per l'ingresso e l'uscita. Si potrebbe impiegare per esempio, la Porta A del PIA come ingresso e la Porta B come uscita. Questo è illustrato in Fig. 8-13. Il programma effettivo diviene quindi:

```
LDAA     SEGNALE A       SEGNALE D'INGRESSO SU A
LDAB     SEGNALE B       SEGNALE D'USCITA SU B
COMA
ANDA     #02
ANDB     #$FD
ABA
STAA     SEGNALE B
```

Il segnale A ed il segnale B rappresentano rispettivamente gli indirizzi della Porta A e della Porta B del PIA. Il segnale è ricevuto nella porta A del PIA, invertito

dal programma e quindi presentato in uscita sulla porta B. Poichè tutti gli altri segnali della porta B non vanno modificati, occorre mascherarli. Questo viene eseguito dall'istruzione AND B mentre ABA ripristina il valore originario degli altri segnali.

Nel caso si dovessero invertire tutte le linee della porta B, il programma sarebbe notevolmente più semplice e non sarebbero richieste operazioni di mascheratura. Questa è una regola generale: ogni volta che un'operazione deve essere eseguita contemporaneamente su otto bit, l'efficienza del programma migliorerà. La lunghezza di un programma non è necessariamente correlata alla funzione da eseguire.

Infine si può notare che dedicare una PIA alla conversione di segnale non è certo il modo più economico per ottenere questa funzione.

FUNZIONI AND-OR REALIZZATE MEDIANTE PROGRAMMA

Una porta AND a 2 ingressi può essere simulata mediante programma come segue:

```
LDAA    SEGNALE A
ANDA    SEGNALE B
STAA    SEGNALE C
```

Anche qui si assumerà, per semplicità, che il segnale A sia formato da otto segnali e così pure il segnale B. L'uscita risultante è il segnale C di otto segnali.

Se i segnali A e B fossero formati da un solo bit, le due posizioni dovrebbero essere corrispondenti.

Analogamente la funzione OR verrà eseguita come segue:

```
LDAA    SEGNALE A
ORAA    SEGNALE B
STAA    SEGNALE C
```

SIMULAZIONE DI FLIP-FLOP

Un flip-flop è caratterizzato da due ingressi e due uscite, $\overline{Q}$ e Q. Da un punto di vista logico un flip-flop può essere rappresentato da un singolo bit che può essere immagazzinato in una locazione di memoria. Un reset ripristinerà un valore 0 nel bit, mentre un set ripristinerà il valore 1. Un programma corrispondente è il seguente:

```
RESET   LDAA    0
        STAA    BASCUL
SET     LDAA    1
        STAA    BASCUL
```

Anche qui il programma opererà contemporaneamente su otto bit. Quindi una parola di memoria conterrà otto flip - flop.

Occorre quindi risolvere il problema seguente: quando dovrebbe essere eseguita una transizione da «0» ad «1» oppure da «1» a «0»? Una transizione si deve verificare dopo che è trascorso un certo *ritardo*. Generiamo quindi dei ritardi.

GENERAZIONE DI UN RITARDO

Un ritardo viene normalmente realizzato da un ciclo di programma. Il tempo di esecuzione di ciascuna istruzione è noto. Se si deve generare un ritardo «T» e se la durata delle istruzioni contenute in un ciclo è «D» occorrerà eseguire il ciclo T/D volte, supponendo che il risultato sia un intero. Poiché qualsiasi ciclo può essere eseguito solo un numero intero di volte, si genererà un ritardo approssimato rispetto a quello richiesto. Alcuni dei microprocessori più recenti sono dotati della possibilità di ritardo programmabile. Una realizzazione di un ritardo con il 6800 consiste nel decrementare un registro un certo numero di volte.

```
            LDA B      VOLTE
CICLO       DEC B
            BNE        CICLO
```

Il numero di volte di esecuzione del ciclo è contenuto nella locazione «VOLTE». Questo viene caricato nell'accumulatore B. Questo registro viene quindi decrementato con l'istruzione DEC fino a che esso raggiunge il valore 0. L'istruzione successiva esegue questa funzione mediante il test del risultato del decremento per controllare se è 0. Finchè non si ha 0 si ritorna all'inizio del ciclo. Quando si raggiunge il valore 0 non ha salto e si passa all'istruzione sequenzialmente successiva del programma. Il ritardo successivo è uguale a VOLTE moltiplicato per la durata di DEC più la durata di BNE. L'istruzione DEC richiede due cicli ovvero due microsecondi e BNE quattro cicli ovvero quattro microsecondi. La durata totale del ciclo è quindi sei microsecondi. Mediante questo programma si possono realizzare ritardi non inferiori di sei microsecondi ed aventi valori pari a multipli di sei microsecondi. Inoltre, essendo impiegato un contatore di 8 bit, il ritardo massimo che si può realizzare è $256 \times 6 = 1536$ microsecondi. Nel caso di ritardi più lunghi esistono diverse soluzioni. Quella più semplice consiste nell'aggiungere istruzioni tra DECB e BNE. Un'istruzione tipica aggiungibile è NOP che introduce un ritardo di due microsecondi. Alternativamente, per avere ritardi più lunghi si può impiegare un registro a 16 bit, come il registro X, come contatore.

LIMITI DELLA LOGICA REALIZZATA MEDIANTE PROGRAMMA

Si è visto come è possibile realizzare mediante programma delle semplici funzioni logiche. Si può ottenere mediante programma anche una combinazione di queste funzioni. Comunque un programma viene eseguito sequenzialmente e non può ese-

guire contemporaneamente più funzioni. In particolare i ritardi normalmente non possono essere realizzati in parallelo (salvo impiegare degli artifici). Nel caso di elaborazione in tempo reale esiste quindi la necessità di introdurre i componenti addizionali per la realizzazione di questi ritardi. In particolare questo è il ruolo del *temporizzatore di intervallo programmabile* (PIT: programmable interval timer), descritto al Capitolo 3. La logica realizzata mediante programma non è volta alla sostituzione dei componenti a livello di singola porta. Essa deve essere impiegata per sostituire componenti a livello *funzionale*. In particolare la disponibilità di un certo numero di altre funzioni all'interno del microprocessore rende possibile la realizzazione di algoritmi molto sofisticati che eseguono le stesse funzioni della logica programmata, ma in modo diverso. L'unico svantaggio significativo è che la logica realizzata mediante programma è più lenta di una realizzazione hardware. Ogni volta che è fondamentale la velocità si deve impiegare il modo più veloce per l'esecuzione di istruzioni programmate: è necessaria la microprogrammazione. Questo si ottiene nei sistemi bit - slice.

Verrà ora considerato un programma completo che realizza una semplice funzione d'uscita.

MUSICA CONTROLLATA DA MICROPROCESSORE

Questo programma genererà note che verranno inviate ad un altoparlante. Si impiegherà semplicemente un microprocessore 6800 dotato di un PIA. Il PIA è connesso ad un altoparlante esterno per mezzo di un resistore (si potrebbe impiegare un amplificatore a transistor).

Si considereranno a questo punto i dettagli della programmazione di un PIA. La Fig. 8-24 e successive riportano la struttura di un PIA. Assumiamo che i bit 4 e 5 del registro di controllo all'interno di un PIA siano posti al valore 1. Il bit 3 del registro di controllo (CRA) porrà quindi il segnale d'uscita CA2 al valore «1» oppure «0». La linea CA2 sarà connessa all'altoparlante. Un suono di una data frequenza verrà generato commutando alternativamente questo bit da «0» ad «1» e da «1» a «0» ad una certa frequenza.

In particolare si assumerà che il PIA sia connesso al bus indirizzi in modo tale che gli indirizzi da 4004 a 4007 corrispondano ai registri interni del PIA. L'indirizzo del registro di controllo per la porta A è 4005. L'indirizzo per il registro d'ingresso-uscita è 4004. Simbolicamente l'indirizzo PIADRA sarà 4004 e l'indirizzo PIACRA sarà 4005.

Per modificare il valore del bit d'uscita su CA2 verrà memorizzato un valore all'indirizzo PIACRA.

Si assumerà che i bit rimanenti dei registri di controllo del PIA siano così posizionati: i bit 012 siano posti ad 1, i bit 6 e 7 siano posti a 0. Non interessano qui i dettagli del controllo del PIA.

Per porre CA2 ad 1 si scriverà in CAR: 00111111.

Per scrivere uno 0 in CA2 si scriverà: 00110111.

Nella rappresentazione esadecimale questi valori valgono rispettivamente: «3F» e «37».

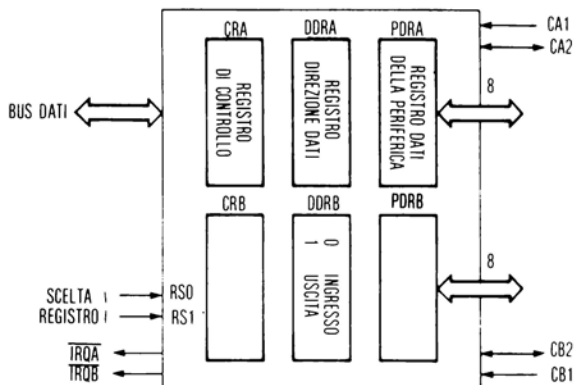
Quindi per generare una transizione alto-basso del segnale d'uscita, si cambierà il valore in PIACRA da 3F a 37 e viceversa.

- OGNI PIA RICHIEDE 4 LOCAZIONI DI MEMORIA (2 PER CANALE)
- LA SCELTA TRA DDR E PDR È ESEGUITA DA 1/0 NEL BIT DI CR "0" È DDR "1" È DR

Fig. 8-24: I 6 registri del PIA.

RS1	RS0	CRA2	CRB 2	REGISTRO SELEZIONATO
0	0	1		REGISTRO A DELLA PERIFERICA
0	0	0		REGISTRO B DELLA PERIFERICA
0	1			REGISTRO A DI DIREZIONE DATI
1	0		1	REGISTRO B DI DIREZIONE DATI
1	0		0	REGISTRO DI CONTROLLO A
1	1			REGISTRO DI CONTROLLO B

Fig. 8-25: Indirizzamento dei registri del PIA.

Questo programma richiede l'impiego di ritardi. Precedentemente è stata descritta una tecnica per generare dei ritardi. Il programma effettivo è il seguente:

```
        ORG     $200
PIADRA  EQU     $4004    Indirizzo del registro d'uscita del PIA
PIACRA  EQU     $4005    Indirizzo del registro di controllo del PIA
        LDA A   # 3F     Valore per commutare ON l'altoparlante
        LDA B   # 3 7    Valore per commutare OFF l'altoparlante
PIANO   STA A   PIACRA   Commuta ON l'altoparlante
        LDX     DELAY    Legge il ritardo
DUREH   DEX              Decrementa
        BNE     DUREH    Ciclo finchè termina il ritardo
        STA B   PIACRA   Commuta OFF l'altoparlante
        LDX     DELAY    Legge lo stesso ritardo
DUREB   DEX              Decrementa
        BNE     DUREB    Ciclo finchè termina il ritardo
        BRA     PIANO    Ritorna indietro per commutare ON
                         l'altoparlante
DELAY   FCB     $80      Ritardo di 1 ms
```

La prima istruzione del programma pone uguale al valore esadecimale «200» l'indirizzo di memoria iniziale. Le due istruzioni successive assegnano il valore esadecimale 4004 e 4005 rispettivamente ai simboli PIADRA e PIACRA. Nelle due istruzioni successive il programma carica i valori esadecimali «3F» e «3 7» rispettivamente in A e B. Il valore del simbolo «DELAY» che appare alla settima istruzione, è definito alla fine del programma come l'esadecimale «80». Si vedrà che questo origina un ritardo di 1 millisecondo. Conseguentemente l'altoparlante genererà una frequenza di 500 Hz.

Si consideri il programma alla label DUREH. Questa è la realizzazione del ciclo di ritardo. Nell'istruzione successiva viene eseguito un test per verificare se il decremento è arrivato a 0. Se si l'istruzione successivamente eseguita è:

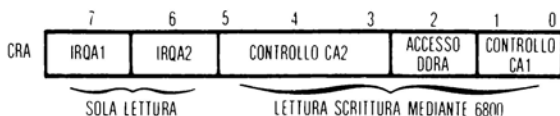
```
STAB    PIACRA
```

Il valore caricato in B (esadecimale «3 7») viene trasferito nel registro di controllo del PIA, commutando off il bit. Successivamente si deve generare un ritardo uguale mentre il bit è off e questo viene realizzato dalle tre istruzioni successive. Al termine del secondo ritardo si ha un salto indietro all'inizio del ciclo generale:

```
BRA     PIANO
```

Verrà generata nuovamente l'oscillazione

CALCOLO DELLA FREQUENZA

Entrambi i cicli impiegati per mantenere il bit al valore 1 oppure 0 sono realizzati in 8 cicli: quattro cicli per DEX e quattro per BNE, cioè otto microsecondi per

DDRA SELEZIONA IL REGISTRO DI INTERFACCIA DELLA PERIFERICA OPPURE IL REGISTRO DI DIREZIONE DATI

IL BIT DI CONTROLLO CA1 SE 0 ABILITA $\overline{IRQA}$

IL BIT DI CONTROLLO SE 1 SELEZIONA LA TRANSIZIONE ATTIVA

CA2 PUO ESSERE CONTROLLATO COME INGRESSO O COME USCITA

Fig. 8-26: Formato della parola di controllo del PIA.

percorrere ogni volta il ciclo globale.

DELAY viene posto uguale a 128 decimale, ovvero 80 esadecimale. Il ritardo risultante vale 128 × 8 = 1024 microsecondi. Approssimativamente risulta 1 microsecondo. Ne risulta un periodo di 2 millisecondi e quindi una frequenza, inverso del periodo, pari a 500 Hz.

Per una frequenza richiesta F qualsiasi, il ritardo necessario verrà calcolato come DELAY = F/4.

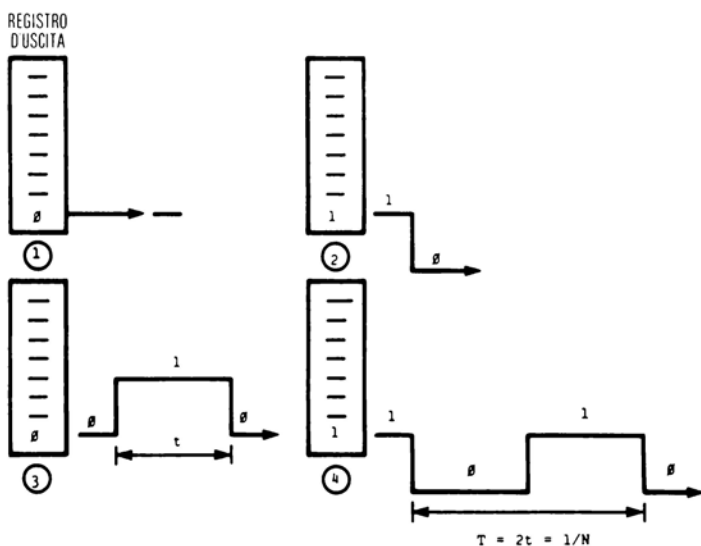



Fig. 8-27: Generazione di un'onda quadra.

La frequenza risultante può quindi essere variata agendo sul ritardo. È quindi molto facile generare un pianoforte elettronico. Questo viene lasciato al lettore come esercizio.

Suggerimento: alla fine del programma, a livello dell'istruzione BRA PIANO, si potrebbe realizzare un salto ad una routine che legge una tastiera. Se viene premuto un nuovo tasto sulla tastiera, questo verrà interpretato come una richiesta di generazione di un nuovo tono. In dipendenza del tasto, verrà ricalcolato il ritardo cioè il ripristino al programma che ritornerà a PIANO per realizzare il codice richiesto. Tutto questo richiede soltanto poche istruzioni addizionali. Le tecniche per la lettura di una tastiera sono state considerate al capitolo relativo alle Interfacce.

VANTAGGI DELLA PROGRAMMAZIONE

I tre vantaggi essenziali della programmazione:

1. **Flessibilità.** In un programma è molto facile introdurre delle variazioni per la disponibilità di potenti strumenti di programmazione che verranno descritti al paragrafo successivo. La ricerca di errori in un programma complesso viene resa notevolmente veloce dalla ricerca di errori in un sistema hardware equivalente. Inoltre, una volta che un programma è stato sviluppato, può essere facilmente cambiato o migliorato. Si è visto, per esempio, che un programma può essere suddiviso in subroutines. Si possono impiegare librerie di subroutines che possono essere convenientemente aggiunte ad un programma. Si possono aggiungere funzioni successive come un algoritmo di ottimizzazione. Tutti i collaudi possono essere fatti da tastiera, con l'assistenza di *debuggers* e di altri programmi. Durante il debugging si possono osservare simbolicamente i dati ed il programma. Il capitolo successivo è dedicato alle fasi ed agli strumenti coinvolti nello sviluppo di un programma.

2. **Velocità.** Si considera la *velocità* a *livello umano* con cui è possibile programmare e sviluppare soluzioni ad un problema assegnato. Non si fa riferimento alla velocità di esecuzione, che in generale è inferiore di quella di una realizzazione hardware diretta. La velocità di sviluppo è particolarmente elevata nel caso in cui si impieghi un programma ad alto livello. Esso consente infatti un accesso veloce e la verifica di nuovi algoritmi. Una volta ottenuta una conoscenza di programmazione a livello di base, si possono sviluppare nuove applicazioni in un tempo molto breve, con una elevata probabilità di successo.

3. **Riduzione dei costi.** Il vantaggio più grosso della programmazione, rispetto alla riduzione dei costi, risiede nel fatto che i programmi sono realizzati su *moduli hardware standardizzati.* Lo stesso modulo può essere impiegato per diverse applicazioni. Si deve aggiungere o rimuovere un numero molto piccolo di componenti per realizzare la scheda per una nuova specifica applicazione. La produzione in massa di tali moduli standardizzati rende possibile costi molto bassi per la parte hardware del sistema. Questa standardizzazione conduce a più bassi costi di documentazione e diminuisce il tempo di debugging.

Questi sono i principali vantaggi della programmazione. Naturalmente esiste un certo numero di problemi coinvolti con la programmazione ma sono stati sviluppati degli aiuti hardware e software per risolvere questi problemi. Il Capitolo 9 è dedicato a questi problemi ed alle cure relative.

Attualmente sono disponibili numerosi *aiuti diagnostici*. In particolare è stato sviluppato un nuovo tipo di strumento, il *microprocessor analyzer*, derivato dall'analizzatore digitale. Questi strumenti sono prodotti, tra gli altri, dalla Hewlett-Packard, Tektronix, Fluke, Biomation. Essi sono indispensabili per una regolazione ed un debugging fine di interfacce hardware complesse. Il loro costo va da 5000 a 15000 dollari.

Esistono diverse alternative a questi strumenti solo nel caso di obiettivi molto limitati. In particolare le schede assemblate sono uno strumento ideale per acquisire familiarità con la programmazione a livello di linguaggio di macchina. Normalmente queste schede non sono dotate nemmeno di un assembler, cosicchè la programmazione deve essere eseguita direttamente in esadecimale. La complicazione risultante limita la lunghezza dei programmi che possono essere ragionevolmente sviluppati su di essi, a poche decine o poche centinaia di istruzioni. La memoria di massa impiegata più frequentemente per tali schede è il registratore a cassette, che può essere facilmente interfacciato alla scheda. Benchè tali schede siano un notevole strumento educativo esse non possono essere impiegate come strumento di sviluppo per un qualsiasi programma «effettivo». Comunque esistono molti sistemi modulati che consentono l'aggiunta di schede fino a realizzare progressivamente un vero sistema di sviluppo. Questa è un'evoluzione interessante. Comunque, ancora una volta, la parte più dispendiosa sarà costituita dalle periferiche e non dalla scheda del processore.

SOMMARIO

Lo sviluppo di un sistema microprocessore coinvolge le tecniche hardware e software. Le tecniche hardware sono state presentate nel corso dei capitoli precedenti. In questo capitolo sono state descritte le tecniche software. Il problema principale, durante lo sviluppo, è costituito dalla fase di debugging software. In questo capitolo è stato discusso a fondo tale problema, assieme agli strumenti disponibili. Si è visto che il sistema di sviluppo costituisce lo strumento più efficiente sia per lo sviluppo hardware che software. Le alternative sono un sistema time-sharing, un computer in-house ed un kit. La caratteristica fondamentale dei sistemi complessi di debugging, particolarmente dei sistemi in tempo reale, è la simulazione in circuito. Infine l'investimento richiesto per un sistema di sviluppo completo, viene abbondantemente ripagato in termini di riduzione del tempo di programmazione e di un completamento più rapido del progetto.

CAPITOLO 9

LO SVILUPPO DEL SISTEMA

Consideriamo, innanzi tutto, la sequenza fondamentale delle fasi necessarie per lo sviluppo di un sistema con microprocessore: considereremo i problemi legati a ciascuna di esse e ne prospetteremo le soluzioni. Descriveremo poi tutti quei dispositivi che sono stati messi a punto per l'attuazione di queste soluzioni nelle varie fasi.

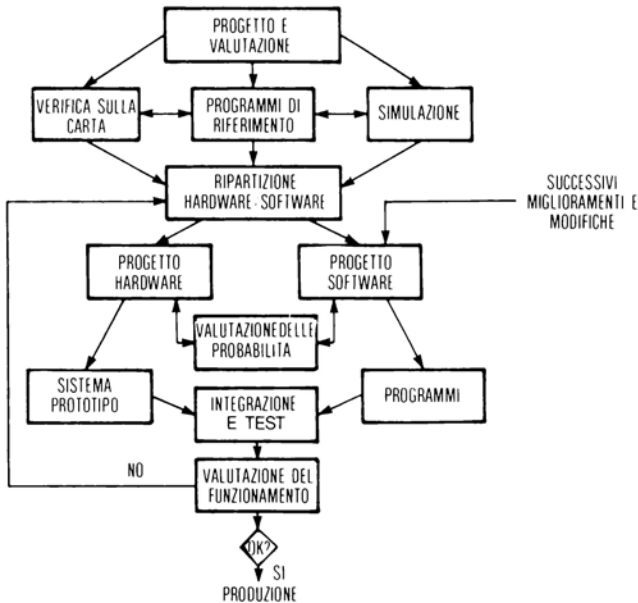



Fig. 9-1: Il diagramma di sviluppo.

La Fig. 9-1 mostra il diagramma di flusso della sequenza delle fasi legate allo sviluppo del sistema: passiamo, quindi, ad esaminarlo, cominciando dall'alto verso il basso: la fase iniziale di qualsiasi sviluppo è, come al solito, una fase di valutazione e progetto, nella quale si prospetta una soluzione al problema. Tale progetto comprende sia la selezione, o assemblaggio, di un sistema completo di microprocessore, sia la messa a punto del software ad esso connesso. La considerazione tec-

nica essenziale, a questo punto, è se il sistema hardware-software che ne risulta soddisfa i requisiti specifici per i quali era stato ideato.

Per verificare le ipotesi e le tecniche di progetto si possono utilizzare tre metodi, separatamente o congiuntamente, come appare nella seconda fila della Fig. 9-1.

La *verifica sulla carta* si riferisce alla prova del sistema che si effettua sulla carta e questo sia per quanto riguarda il progetto logico che per i programmi. Per fare un esempio, l'esercizio che era stato raccomandato alla fine della routine di moltiplicazione, nel precedente capitolo, è un esercizio di verifica sulla carta, dove il lettore esegue il programma a mano e riempie le voci in una tabella corrispondente ai valori dei registri. Il progettista può allora assicurarsi se i risultati del suo programma sono corretti. È chiaro che questo è il metodo più economico, ma è anche il più lungo e laborioso. Questo metodo viene normalmente realizzato a livello di diagramma per verificare la correttezza complessiva del progetto, ma non darà una valida valutazione sul suo funzionamento: a questo proposito, per una miglior valutazione, bisogna ricorrere all'automazione.

I *programmi di riferimento* (benchmark programs) sono dei programmi specifici che vengono scritti da chi li utilizza per valutare l'efficienza di un dato sistema di microprocessore; spesso, sono gli stessi costruttori che li forniscono già pronti. Sono, senza dubbio, dei programmi che realizzeranno, per esempio, un trasferimento di blocco od una serializzazione, tuttavia, essi furono *scritti dal costruttore* e ottimizzati con molta cura: noi, qui, affermiamo che essi non sono dei *programmi validi.* Un programma di riferimento valido deve essere scritto da colui che lo usa ed il programma finale da eseguire sul sistema dovrà essere scritto da lui stesso. Ogni programma di riferimento dovrebbe, perciò, rappresentare la sua abilità di programmare, se questa significa fornire una misura reale dell'efficienza dei suoi programmi.

Una seconda difficoltà consiste nel fatto che la maggior parte delle applicazioni non può essere caratterizzata da tali semplici programmi e richiede l'esecuzione di un insieme caratteristico di istruzioni: sfortunatamente, questo non esiste ed è molto difficile decidere se un programma più lungo abbia dei valori di riferimento reali.

Il vantaggio del programma di riferimento consiste nel fatto che, quando è definita chiaramente la sua applicazione, è possibile eseguire tale programma di trasferimento di blocco sull'8080, sul 6800, sul 2650 e, quindi, decidere quale di essi sia il più indicato per tale applicazione. Purtroppo una tale definizione chiara è piuttosto rara.

Un terzo metodo possibile è la *simulazione*, che può essere realizzata in vari modi. La simulazione di programmi può essere svolta scrivendo quest'ultimi in un linguaggio ad alto livello e facendoli eseguire in procedimento simulato su una grossa macchina; ciò è costoso, necessita di molto più tempo, ma dà dei risultati estremamente precisi. È chiaramente il metodo più preciso di tutti.

Per concludere, e come al solito, sarà l'esperienza uno dei principali criteri usati

dal progettista per valutare se un determinato progetto soddisfa o no le specifiche richieste: i tre metodi descritti serviranno come possibili strumenti.

Passiamo ora ad esaminare la terza riga orizzontale della Fig. 9-1: La ripartizione hardware - software.

Un'altra delle decisioni critiche che devono essere prese dal progettista riguarda quanto deve essere realizzato in forma di chips e quanto in forma di programmi, la cosiddetta «ripartizione hardware-software». Il criterio essenziale, a questo punto, è la *quantità* di sistemi prodotti: se la quantità dovesse risultare troppo grande, il numero dei componenti hardware deve essere accuratamente ridotto al minimo e nel limite del possibile si cercherà di sopperire con il software. Se invece si deve produrre un numero limitato di sistemi, risulta di solito vantaggioso usare un numero maggiore di chips hardware se questo può ridurre la complessità della programmazione; a questo punto, è solo questione di valutare la complessità del software, il tempo necessario ed il costo. Se la riduzione del costo del software ed il tempo impiegato sono significativi, allora vale la pena di aggiungere un 10 o 15 dollari di chips in più, e questo è ciò che normalmente accade.

La ripartizione hardware-software è uno dei compiti più delicati che il progettista deve affrontare: tale distribuzione dovrebbe essere costantemente valutata e rivalutata nelle varie parti del progetto, così come dovrebbero essere considerate con cura le probabilità. La ripartizione ha un'influenza maggiore sul progetto software.

Una volta che queste due parti sono state ben ripartite, si possono condurre i progetti parallelamente.

Sia il progetto hardware che quello software possono essere realizzati in parallelo: questa è la principale differenza che scaturisce quando si confrontano lo sviluppo di un microprocessore con quello di un hardware normale. Il progetto hardware può essere realizzato indipendentemente dallo sviluppo software. Il primo è di solito semplice quando si tratta di un sistema di microprocessore standard, ma può divenire più complesso quando si tratta invece di interfacce insolite. Normalmente, il compito più significativo è il progetto software e, come verrà mostrato nei paragrafi seguenti, si sono introdotti un certo numero di strumenti di sviluppo e di sistemi tali da rendere possibile una sua realizzazione più efficiente, indipendetemente dall'hardware.

Durante tutto il progetto, si dovrebbe portare avanti una valutazione delle probabilità (linea 5 di Fig. 9-1), al fine di riesaminare la ripartizione hardware/software che era stata fatta all'inizio e così facendo può sorgere la necessità di passare da un polling ad un interrupt, per esempio, od all'aggiunta di alcuni codificatori hardware per semplificare il progetto software o per migliorarne la prestazione.

A progetti ultimati, ne risultano rispettivamente un prototipo di sistema hardware ed una serie di programmi che si auspica siano corretti.

La fase successiva è l'integrazione e il test (linea 6 di Fig. 9-1).

L'*integrazione* consiste nell'inserire i programmi software nel sistema prototipo e nel rivedere il sistema hardware/software che ne risulta per poterne eliminare gli errori (debugging). Questa è, spesso, la parte più complessa e col maggior dispendio di tempo di qualsiasi sviluppo di sistema: è la cosiddetta fase dell'«indice puntato» nella quale i progettisti dell'hardware e del software si puntano gli indici accusandosi vicendevolmente: «è colpa tua».

Questa fase mette, allora, in risalto l'importanza di un progettista unico, esperto in entrambe le progettazioni, che possa risolvere queste controversie.

Per facilitare l'integrazione finale ed il test dei sistemi reali è stato introdotto uno strumento essenziale, l'*emulatore in circuito* (ICE), che verrà descritto alla fine di questo capitolo.

Alla fine, si è messo a punto un sistema realizzato con un hardware ed un software privi di errori. A questo punto, il sistema dev'essere sottoposto ad una valutazione della sua prestazione ed è auspicabile che essa soddisfi le specifiche e possa andare in produzione. Nel caso non soddisfi tali specifiche deve ritornare alla fase di progetto o, meglio, alla fase di ripartizione. Normalmente è possibile individuare una o più funzioni che non rispondono alla prestazione richiesta ed, allora, i programmi software verranno normalmente convertiti in componenti hardware addizionali per migliorarne la velocità.

Bisogna precisare che, in un buon progetto, non ci dovranno essere cambiamenti nell'hardware e qualsiasi ulteriore miglioramento o modifica dovrà essere fatto aggiungendo delle nuove funzioni software come indicato dalla freccia sulla destra in Fig. 9-1.

SVILUPPARE UN PROGRAMMA

Lo sviluppo di un programma comporta la codifica di un algoritmo in un linguaggio di programmazione, come già abbiamo visto nel precedente capitolo. Il problema essenziale diventa il *debugging* del programma, ovvero, verificare che il suo funzionamento è corretto. Passiamo ora ad esaminare le operazioni che esso comporta ed il supporto richiesto, riferendoci alla Fig. 9-2: le apparecchiature appaiono sulla sinistra, i programmi appaiono sulla destra della figura, mentre nella colonna centrale sono rappresentanti il processore o i processori necessari al debugging del programma.

Il programma scritto a mano deve essere inserito, prima di tutto, nella memoria del sistema per l'esecuzione del debugging: è questa la fase 1 nella quale l'utente scrive il suo programma alla tastiera, che verrà immesso nella memoria del sistema (e appare sulla destra dell'illustrazione) come il *programma sorgente*, scritto in forma simbolica. Questo programma potrà essere scritto in un linguaggio ad alto livello o in linguaggio assembler.

Per facilitare l'ingresso del programma nel sistema è auspicabile avere a disposizione un *editor*, come appare nel cerchio sulla destra della figura. L'editor è uno speciale programma che permette un'opportuna manipolazione del testo; con esso diventa possibile fare cose come: «torna alla linea 3 ed inserisci ciò che segue» oppure: «cerca R2 nel testo e sostituiscilo con R3»; oppure: «aggiungi ciò che sto per scrivere dopo la linea 42», o «... prima della linea 3».

Un editor potente è essenziale per la velocità con cui un programma può essere scritto nel sistema e convenientemente modificato una volta che l'errore è stato individuato. Dopo aver inserito tutto il programma nel sistema è, di solito, conveniente stamparlo per potersi rendere conto che è completo e corretto; si usa allora una stampante (fase 2 di Fig. 9-2). Il programma sorgente è stampato dalla stampante ed appare sotto forma di *listing*. Presumendo che il programma sia corretto, vogliamo ora procedere con l'esecuzione: ora, siccome il programma è scritto in codice sorgente, vale a dire in forma simbolica, esso deve essere tradotto in forma tale da poter essere elaborato dalla macchina.

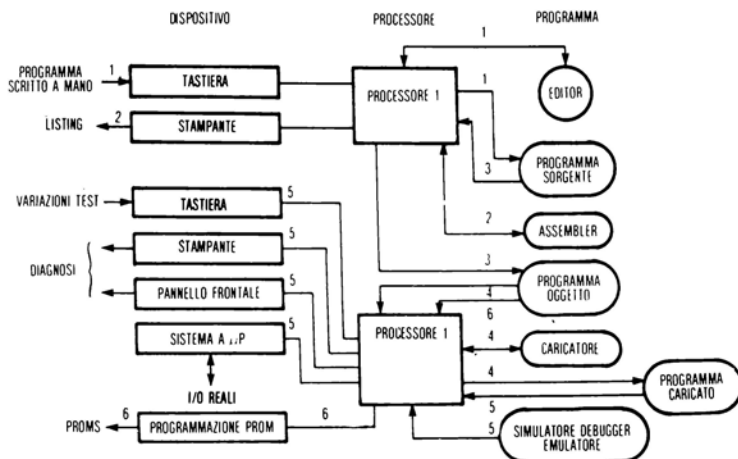



Fig. 9-2: Le caratteristiche del software.

L'*assembler* svolge questa funzione: esso traduce il codice sorgente in programma oggetto binario che può essere elaborato direttamente dalla macchina. L'assembler sostituirà gli indirizzi reali a quelli simbolici, la codificazione binaria reale ai nomi simbolici e l'appropriato codice binario di istruzioni al loro codice mnemonico. È questa la fase 3 in cui si è creato il *programma oggetto.*

Il programma oggetto è una sequenza di parole binarie, le quali possono essere ora elaborate direttamente dal processore; a questo punto, si rende necessario ese-

guirlo sul processore reale (o anche su uno simulato).

Da questo nasce una considerazione interessante: non era necessario usare il microprocessore originale per sviluppare il programma oggetto; tutte le funzioni che venivano richieste al processore 1 consistevano nel fornire i mezzi per l'editing e l'assembly e qualsiasi processore può fare ciò. Infatti, usando un processore più grande ne risultano dei vantaggi maggiori: disponibilità di periferiche costose e potenti, velocità nell'elaborazione e mezzi software sofisticati. In quel caso, l'assembler è chiamato *cross-assembler.*

Un *cross-program* è un programma per la macchina A che risiede nella macchina B; un cross-assembler per un Motorola 6800 è un assembler che produrrà un codice 6800, ma che lo esegue su un Digital Equipment PDP11.

Procediamo, ora, con l'esecuzione vera e propria: se il processore 1 era in realtà il vero microprocessore, si dovrà continuare ad usarlo; se era un altro computer dovremmo, a questo punto, deviarlo verso il processore finale. Vedremo più tardi che c'è un'alternativa e che il processore 1 poteva essere mantenuto in funzione se si disponeva di un *simulatore.*

Il programma oggetto deve essere ora sistemato nella memoria del sistema nel quale verrà eseguito: è questa la fase del *loading* (caricamento) (fase 4, Fig. 9-2), che viene eseguito da un programma caricatore (*loader*). Ne risulta un programma caricato, od *assoluto*, che risiede nella memoria e che può essere, quindi, eseguito. Siamo, ora, nella fase 5: l'esecuzione reale, che procederà sotto il controllo di un *debugger.*

Il *debugger* è un programma speciale che esegue il debugging finale di un programma utente. Esso offre delle opportunità quali: «fermati all'istruzione»; «esegui le istruzioni una alla volta»; «visualizza i contenuti dei registri»; «visualizza i contenuti della memoria (in binario, esadecimale o anche in forma simbolica)». Usando un debugger, i contenuti dei registri possono essere cambiati ed il programma può ricominciare daccapo. Il vantaggio di questa opportunità può anche non apparire ovvio all'inizio, ma proviamo a pensare come sia possibile esaminare i contenuti dei registri di un microprocessore hardware una volta che esso si è formato: è impossibile! L'unico modo sarebbe di collegare delle microsonde ai flip-flop reali dei registri *all'interno dei chip* ma, una volta che il microprocessore è venduto, si possono vedere solo i contenuti dei suoi bus. Una caratteristica essenziale di un debugger è quella di permettere l'esame dei contenuti dei registri e di cambiarli e questo viene fatto o eseguendo delle istruzioni di visualizzazione sul microprocessore od eseguendo sotto il controllo di un simulatore o di un emulatore, i quali memorizzino una copia del valore dei registri nella memoria.

Ogni qualvolta non si esegue il programma sul microprocessore reale, si utilizzano un *emulatore* od un *simulatore.* Un *simulatore* è un programma che scorre su un'altra macchina, diciamo un IBM 370, ed esegue un codice 8080: esso simula un

8080. Naturalmente, un simulatore non può operare in tempi reali a causa della lentezza dell'interpretazione software interessata, mentre un emulatore è un simulatore che opera in tempo reale. L'*emulazione* implica che il comportamento è identico all'originale. Di ciò verranno fatti alcuni esempi più avanti in questo stesso capitolo.

Mentre il programma viene debugged sotto il controllo del debugger, viene compilata una diagnosi che normalmente verrà stampata sulla stampante o visualizzata su un pannello frontale od altro mezzo. Sono allora possibili delle piccole correzioni se l'utente vuole modificare direttamente le posizioni della memoria, altrimenti egli deve ritornare all'inizio del programma, inserire le correzioni, assemblare, e far scorrere di nuovo l'intera sequenza.

Un'operazione essenziale di un debugger è di fornire i *break-points* (punti di arresto), che sono degli indirizzi specificati dall'utente e indicano dove il programma dovrà fermarsi automaticamente. L'utente può allora esaminare il valore delle variabili nella memoria od i contenuti dei registri e l'esecuzione del programma è per quel momento sospesa.

Uno dei classici debugger fu sviluppato dalla Digital Equipment per la sua linea PDP: si tratta naturalmente del «DDT».

Dopo che il programma è stato completamente debugged, cioè si presume che dopo la sua esecuzione sia corretto, il codice oggetto deve essere inserito nella PROM vera e propria. Durante il debugging, esso risiedeva in una RAM ed a correzione ultimata viene inserito nella PROM: un programmatore PROM sarà collegato al sistema di sviluppo (fase 6 di Fig. 9-2) ed i contenuti binari del programma verranno trasferiti nei chips della PROM, i quali saranno, quindi, solo inseriti nel sistema e ci si augura funzionino bene. Sono disponibili anche dei programmatori PROM soltanto e sono illustrati alla fine del capitolo.

Esaminiamo, ora, i mezzi necessari allo sviluppo di un programma efficiente. Tutte le attrezzature software sopra citate dovrebbero essere a disposizione per uno sviluppo appropriato di qualsiasi programma che comprenda un numero ridotto di istruzioni, ma diventano una necessità assoluta per tutti quei programmi con centinaia, o migliaia di istruzioni.

L'impiego di questi programmi di supporto può non essere semplice, a meno che si disponga di ulteriori attrezzature; in particolare essi dovrebbero essere direttamente accessibili, memorizzati in un mezzo appropriato per essere accessibili al processore. Questo comporta la necessità di un *file-system* generale, il cui supporto può essere costituito da *nastri a cassette* o da floppy-disk. I primi offrono il vantaggio di un costo molto basso, però sono piuttosto lenti: occorre più di un minuto per poter accedere all'informazione desiderata e tale accesso è sequenziale, per cui se si deve accedere a dei dati in differenti punti del nastro occorrono dei lunghi avvolgimenti e riavvolgimenti, che costituiscono delle perdite di tempo. L'alternativa è rappresentata dai *floppy-disks*, che sono in definitiva il migliore mezzo disponibile per

la memorizzazione, nonostante il loro costo sia più elevato. Un floppy consente il rapido accesso a qualsiasi parte del disco nel giro di pochi millisecondi ed è importante sottolineare che in mancanza di tali dispositivi, lo sviluppo può essere abbastanza sgradevole. Supponiamo che si stia sviluppando un programma di 2K e di non avere a disposizione tale file-system: il programma utente così come i programmi di supporto saranno memorizzati su un nastro perforato; perforare un programma può richiedere 20 minuti o più sulla teletype. Gli altri programmi di supporto necessari per il suo debugging dovranno, quindi, essere memorizzati, uno alla volta, nella memoria del sistema e ciò richiede 10 o 20 minuti ogni volta. È chiaro che tutto questo rappresenta un ostacolo seccante a qualsiasi sviluppo.

SCELTE BASILARI PER LO SVILUPPO DEL SISTEMA

Le tecniche per lo sviluppo del sistema sono state presentate ora a livello di hardware, di software ed a livello di sistemi; le scelte fondamentali da fare sono:

1) scegliere il microprocessore;
2) eseguire la ripartizione hardware/software;
3) scegliere il linguaggio di programmazione;
4) scegliere gli strumenti di sviluppo necessari.

La *scelta di un microprocessore* è già stata trattata nel Capitolo 4; per riassumerla brevemente, si devono tener presenti due criteri fondamentali che sono la quantità e la qualità di un prodotto che abbia una prestazione sufficiente per l'applicazione a cui è destinato. Di solito, le ulteriori considerazioni riguardano la disponibilità di tutti i componenti ed il supporto di sviluppo.

La *ripartizione hardware/software* rimane una valutazione del progettista sui rispettivi pregi costo/prestazione delle tecniche da lui prescelte. In questo libro sono state presentate tutta una serie di tecniche usate per assemblare un sistema a microprocessore ed una serie di tecniche software essenziali per la sua realizzazione: esse dovrebbero costituire un valido aiuto in questa scelta.

Passiamo ad esaminare le ultime due scelte: un *linguaggio di programmazione* e gli *strumenti di sviluppo adatti.*

Sono tre le scelte fondamentali per un *linguaggio di programmazione*:
1) binario diretto o esadecimale;
2) linguaggio assembly;
3) linguaggio ad alto livello.

La *programmazione in binario*, o nell'equivalente esadecimale, è una scelta un poco misera in quanto, chiaramente, non richiede nessun supporto in termini di hardware e di software: questo metodo si usa sui sistemi più semplici quali quelli a scheda singola e sulle attrezzature didattiche. L'utente comunica col sistema tramite una tastiera esadecimale e da quattro a sei LED: le istruzioni ed i dati possono, quindi, essere immessi nel sistema per mezzo dei tasti in forma esadecimale. Se dal

punto di vista dell'hardware ciò risulta economico, da quello della programmazione esso risulta lento e gravoso. È vero che è possibile progettare dei programmi corti ed inserirli nella macchina con questo sistema, ma è assolutamente sconsigliabile farlo per programmi più lunghi, anche perchè si può disporre di altre alternative.

Questo metodo viene anche usato durante la fase finale del debugging, quando, cioè, l'utente vuole cambiare solo una od alcune delle istruzioni all'interno della memoria durante il procedimento di debugging; tutto questo gli risparmia l'incomodo di dover ritornare all'editor per poter inserire le variazioni, riassemblare, caricare e tornare di nuovo al procedimento di esecuzione.

In breve, la programmazione in binario od in esadecimale è usata soltanto per programmi brevi, quando non esistono altre alternative. L'attitudine costante dell'evoluzione dei computer è stata quella di realizzare degli ausili di programmazione sempre più potenti, senza doverli complicare troppo per l'utente; sarebbe stato irragionevole non considerare anche le sue esigenze, a meno che non ci sia stato un costo proibitivo.

Il *linguaggio a livello assembly* è semplicemente la rappresentazione mnemonica o simbolica del binario. In termini di efficienza del programma utente, è senz'altro il metodo migliore da usarsi ed è uno di quelli usati più frequentemente nelle applicazioni industriali. Sfortunatamente, la programmazione con questo linguaggio richiede la manipolazione di registri, di bus e di testing bit, oltre ad una buona conoscenza della struttura hardware del sistema che risulta in un'ottimizzazione di queste risorse. Tutte le volte che si desidera una prestazione del software, è questa la tecnica più indicata. L'assembler si prende cura di convertire automaticamente i programmi simbolici in forma binaria tale da poter venire elaborata ed, inoltre, esso rivelerà gli errori di sintassi grossolani e li segnalerà all'utente prima che entrino in esecuzione. Esso, naturalmente, non rivelerà mai gli errori logici. Lo svantaggio consiste nella programmazione noiosa con linguaggio assembler e nel tempo che essa richiede.

Di un *linguaggio ad alto livello* si è già parlato nel capitolo precedente ed è del tipo «PL/M» o «BASIC» che consente al programmatore delle istruzioni potenti per stabilire il suo algoritmo. Questo linguaggio è di solito legato alla convenzione di utilizzarlo nella determinazione delle specifiche dell'algoritmo (matematico, finanziario, ecc.). È, quindi, possibile codificare, cioè programmare, l'intero algoritmo in un tempo breve: la programmazione con questo linguaggio è almeno dieci volte più veloce che quella in assembler, specialmente per i programmi lunghi. Lo svantaggio consiste nel fatto che il compilatore, che produce il codice oggetto, è inefficiente: esso compila un'istruzione ad alto livello in un certo numero di istruzioni binarie a livello di macchina ed a causa della non ottimizzazione dell'uso dei registri, causerà molti trasferimenti di registro non necessari. Normalmente, un compilatore produrrà un numero di istruzioni da due a cinque volte maggiore di quello che produrrebbe un programmatore a livello assembler molto capace, risultandone uno spreco

di memoria ed un tempo di esecuzione da due a cinque volte più lento e peggiore. Tuttavia, ciò non può rappresentare un'obiezione, bisognerebbe allora considerare due punti essenziali:

a) Se il programma è molto complesso, in relazione all'esperienza del programmatore, è possibile che non venga mai realizzato correttamente se esso deve essere programmato in linguaggio assembler; al contrario, programmandolo in linguaggio ad alto livello è probabile che sia scritto e debugged rapidamente. Successivamente, è possibile produrne una versione ottimizzata a mano. I linguaggi ad alto livello rendono possibile la stesura rapida di programmi corretti anche da parte di programmatori non esperti; infatti, questi linguaggi sono utilizzati dalla grande maggioranza di quegli appassionati che non si preoccupano tanto dell'efficienza del programma quanto di realizzare le sue funzioni rapidamente ed in modo corretto;

b) nel caso di programmi molto complessi, la complessità di codificarli in linguaggio assembler può essere tale da sconsigliarne la realizzazione in modo assoluto, con il rischio, poi, che sia del tutto inefficiente: la sola soluzione ragionevole diventa il ricorso all'uso del linguaggio ad alto livello. Si potrà, allora, dire che se il programma è di tale complessità, il costo di programmazione sarà molto alto e può non essere ragionevole l'uso di un microprocessore.

Ci siano consentite, a questo punto, un paio di considerazioni. È possibile codificare la soluzione di un problema in un linguaggio ad alto livello in maniera rapida ed efficiente, specialmente se si usa un *interprete interattivo*, quale è il BASIC. Un interprete interattivo può eseguire le istruzioni una alla volta e verificare immediatamente se la sintassi è corretta e se non è stato commesso nessun errore. Un interprete traduce ciascuna istruzione in binario immediatamente, così che ogni parte del programma può essere eseguita non appena essa viene stampata e fornisce una diagnosi preziosa ed efficace, risultandone un debugging più rapido.

Inoltre, molti compilatori consentono all'utente di sviluppare prima un programma in linguaggio ad alto livello e, sostituire poi, i moduli a linguaggio assembler dentro i loro programmi ad alto livello. Il vantaggio è evidente: se la prestazione non è sufficiente o lo spreco di memoria è troppo rilevante, l'utente può allora codificare, sezione per sezione, i suoi moduli ad alto livello in linguaggio assembler e sostituirli progressivamente nel suo programma.

Per fare un esempio, è possibile codificare rapidamente un programma in linguaggio ad alto livello e farlo eseguire correttamente; a questo punto, il prototipo è ultimato ed il software è subito disponibile, il che prova che il sistema soddisferà senz'altro il compito richiestogli. Questo di solito si traduce in un premio per la persona responsabile del progetto ed allora, mentre si approntano i piani per la produzione, la stessa persona ricodificherà direttamente in linguaggio assembler tutti i moduli che siano necessari. È stato provato che la struttura complessiva del programma è corretta e diventa, perciò, un semplice compito di codificazione. Come

risultato, l'efficienza del programma sarà migliorata di molte volte e l'uso della memoria da parte del programma sarà ridotto conseguentemente. Risultato: un prodotto più veloce, migliore, più economico e, di solito, un secondo aumento di stipendio. Un altro vantaggio di questa strategia è di introdurre per primi il prodotto nel mercato e battere la concorrenza. Il prodotto prima di entrare nella produzione di massa, viene ottimizzato.

Il metodo razionale per decidere se usare un linguaggio ad alto livello o un'assembler è di valutare la complessità e l'esperienza disponibile e, quindi, stimare il costo del software che ne risulta; se se ne producono poche unità, questo costo sarà prevalente.

Il costo del software potrà essere notevolmente ridotto passando ad un linguaggio ad alto livello, col risultato di un costo di memoria aggiuntivo per ogni sistema, ciononostante, il costo aggiuntivo per l'hardware sarà ampiamente compensato da un risparmio del tempo di programmazione.

Dall'altro lato, se il sistema deve essere prodotto in grande quantità, è maggiormente rilevante la quantità di memorie utilizzate: i costi di programmazione verranno distribuiti su un numero elevato di unità.

L'idea, tuttavia, è di valutare il costo di programmazione e di dividerlo per il numero di unità previsto, dando luogo ad un costo software per unità, che può essere raffrontato al risparmio nell'hardware per unità, realizzato usando il linguaggio assembler.

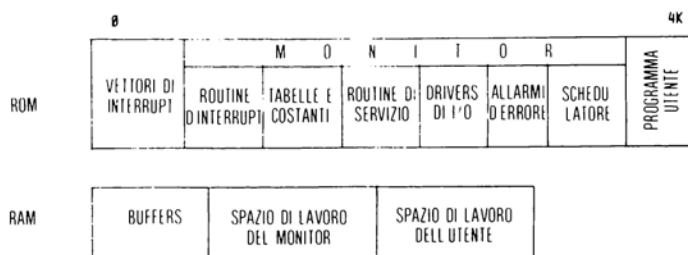



Fig. 9-3: La mappa caratteristica di memoria.

GLI STRUMENTI DI SVILUPPO

Una delle tecniche di sviluppo di programma che abbiamo presentato non richiede ovviamente nessuno strumento particolare: si tratta della programmazione a mano. Anche quando un programma viene sviluppato direttamente in binario od esadecimale, tutto ciò che si richiede è una tastiera esadecimale e un LED per visualizzare la risposta, per il resto occorrono solo carta e matita.

Le altre alternative, più comunemente usate, richiedono degli strumenti potenti se si vuole realizzare più facilmente la programmazione. Ci sono tre strumenti disponibili e ne presenteremo e valuteremo i rispettivi pregi; essi sono: il sistema di time - sharing, l'in - house computer e un sistema di sviluppo.

Il sistema di time - sharing

Il sistema di *time-sharing* consiste in un computer generico che un certo numero di utenti usano contemporaneamente, in un procedimento interattivo (il sistema risponde immediatamente in linea). Il servizio offerto da questo sistema è il migliore che si può ottenere: il dialogo con la macchina è istantaneo, senza attese, ed è, di solito, equipaggiato con periferiche molto sofisticate e supporto software, con potenti file - system, editors ed altre attrezzature. Un programma di microprocessore può essere quasi completamente sviluppato e debugged su tale sistema, il quale sarà equipaggiato con un cross-assembler od un cross-compiler per il 2650, l'8080, il 6800 o qualsiasi altra MPU. Oggigiorno, praticamente tutti i sistemi in commercio forniscono i cross-programs necessari per generare il codice di qualsiasi dei microprocessori esistenti: i programmi utenti possono, quindi, essere generati e debugged sul sistema in maniera molto efficiente, ma per eseguire i programmi bisogna ricorrere ad un simulatore ed è questo l'inconveniente di questo sistema.

Quando i programmi vengono eseguiti da un simulatore, essi possono essere solo simulati e, mentre è possibile debug la logica del programma, non è possibile verificare la prestazione input/output e misurare la sua velocità. In realtà, si può conteggiare la durata simulata delle istruzioni che verranno eseguite dalla MPU, tuttavia, è molto difficile simulare la struttura del sistema, in modo particolare le funzioni input - output che verranno eseguite.

Abbiamo anche dimostrato che i sofisticati chips di input/output che sono stati aggiunti sono diventati parte del processore e sono anche programmabili; dovrebbe essere necessario simulare anche loro, per poter verificare i risultati completi dell'esecuzione del programma.

Un sistema di time-sharing offre anche il vantaggio di consentire l'ingresso contemporaneo di parecchi utenti, usando altrettanti terminali. A prima vista questo può non apparire un vantaggio: per dei piccoli sviluppi può essere sconveniente avere troppe persone che lavorano simultaneamente sullo stesso programma.

Per concludere, il sistema di time-sharing è lo strumento migliore di cui si possa disporre per lo sviluppo veloce del programma stesso, tuttavia esso non permette il suo debugging dal punto di vista della determinazione del tempo, la verifica delle funzioni input - output ed è anche piuttosto costoso.

In - House Computer

Qualsiasi computer può essere usato per eseguire i cross - programs. Quando si

dispone di un computer in-house, è, in realtà, possibile eseguire il cross-assembler, il cross-compiler ed altri programmi su di esso, tuttavia il servizio fornito sarà buono tanto quanto quello in-house. Se il computer è usato nel procedimento di gruppo, cioè se esso è vincolato a schede, nastri di carta od altri mezzi ed i risultati arrivano dopo ore o giorni più tardi, può essere l'ostacolo maggiore allo sviluppo del programma; di solito, lo sviluppo di un programma eseguito da un microprocessore comprende variazioni a livello di bit e diventa ben presto intollerabile sopportare dei ritardi di ore o giornate per eseguire qualsiasi variazione.

Tuttavia, se un in-house computer dispone di capacità di time-sharing, i suoi vantaggi sono gli stessi spiegati in precedenza ed, inoltre, ha il vantaggio di essere normalmente libero.

Infine, un sistema in-house, proprio come un sistema di time-sharing, non consente la possibilità di verifica hardware o l'integrazione del sistema. Il solo strumento che consente sia il debugging del software che dell'hardware è il sistema di sviluppo che passiamo ora ad esaminare.

Il sistema di sviluppo

Il sistema di sviluppo è un microcomputer equipaggiato con tutti i mezzi richiesti per lo sviluppo di un sistema pratico. Il suo aspetto esteriore assomiglia a quello di

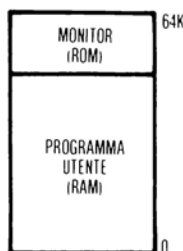



Fig. 9-4: L'uso di memoria in un sistema di sviluppo.

un minicomputer tradizionale, cioè una scatola, ed è equipaggiato con un unico pannello frontale per facilitare il debugging. Di solito, il costruttore che vende il sistema di sviluppo usa il suo microprocessore, ma questo non è necessario; infatti, c'è una gran varietà di tali sistemi in commercio che usano il microprocessore per produrre il codice per i microprocessori X, Y o Z. Il sistema di sviluppo dovrebbe offrire tutti i mezzi che avevamo ritenuto indispensabili, quali un file system, p.e., in grado di collegare facilmente una varietà di periferiche (un disco, p.e., richiede un Disk-Operating System o DOS). Dal punto di vista del software esso dovrebbe essere equipaggiato con tutti i programmi richiesti, quali l'editor, l'assembler, il debugger ed altri programmi di supporto, quali interpreter a pannello frontale e routi-

nes di servizio per il debugging simbolico. È anche auspicabile avere un compilatore locale (PL/M) o in interpreter (quale il BASIC), se esso è progettato per programmare in linguaggio ad alto livello. Sfortunatamente, questi compilatori richiedono una grande quantità di memoria e raramente sono realizzati sul sistema di sviluppo stesso.

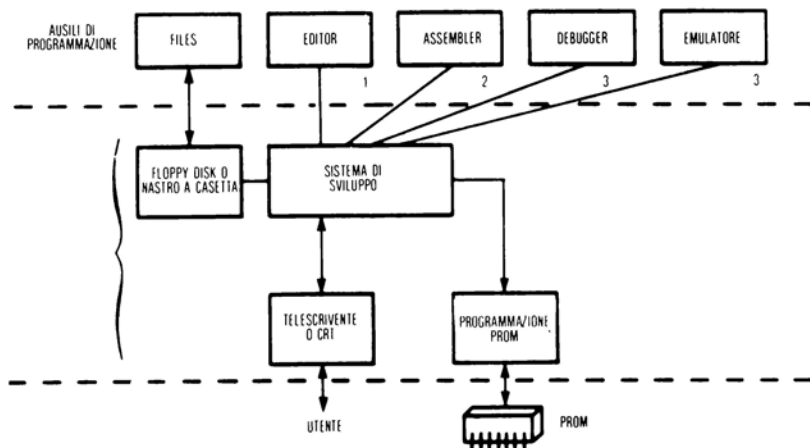



Fig. 9-5: Il sistema di sviluppo.

Esaminiamo ora alcuni sistemi reali:

«*The Exorcizer*» della Motorola (Fig. 9-6) come qualsiasi altro sistema di sviluppo ha l'apparenza di una scatola di minicomputer con un pannello frontale ridotto ed include tre moduli base:

- un modulo di processore;
- un modulo di debugging chiamato EXBUG;
- un modulo di baud-rate, che consente il collegamento alle periferiche con una velocità regolabile da 110 a 9600 baud.

La scatola può contenere fino a 14 moduli. Per ogni sviluppo di sistema pratico sarà necessario aggiungere moduli di memoria quali RAM, ROM, oltre a delle interfacce di input-output da inserire nel sistema. Bisogna sottolineare che il costo della scatola così come è venduta non rappresenta il costo più rilevante del sistema: si dovranno aggiungere allo chassis un certo numero di *schede* e, ancora più importante, le varie *periferiche*, quali:

- un dispositivo di entrata, quale una tastiera od una teletype;
- un dispositivo di uscita, quale un video CRT;
- un dispositivo stampante per copie stampate come la stampante ad alta velocità;

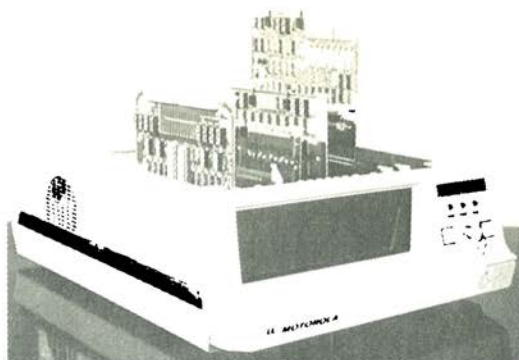



Fig. 9-6: L'Exorcizer della Motorola.

- una memoria di massa per memorizzare i files, quale i dual-floppies (sono necessari due dispositivi per i files accoppiati, come quando vengono messe in ordine alfabetico)
- infine per produrre le PROM, un programmatore PROM.

Sarebbero consigliabili anche altri dispositivi quali un *in-circuit emulator*. Il costo della scatola di per se stesso (The Exorcizer) è basso (circa 2500 dollari), ma una volta che si aggiungono i costi delle varie periferiche ne risultano delle cifre attorno ai 10-15.000 dollari, che è la spesa reale di un sistema veramente utile, perchè qualsiasi risparmio realizzato sulle periferiche verrà tradotto in inconvenienti nella programmazione e di conseguenza di solito, delle perdite finanziarie ancora maggiori in termini di tempi di programmazione e di ulteriori aggravi.

L'*Intel MDS80* (Fig. 9-7) ha anch'esso le sembianze di una scatola di minicom-

Fig. 9-7: L'Intel MDS.

puter e contiene una scheda 8080 CPU, una 16K RAM, una 2K ROM ed una 262 byte PROM, oltre alle interfacce periferiche per una TTY, un CRT, una stampante di linea, un PTR ed un PTP ad alta velocità ed un programmatore PROM.

Il monitor utilizza la 2K ROM; un assembler locale richiede una 12K RAM ed un editor richiederà una 8K RAM, incluso uno spazio di lavoro di 4,5K. Questo significa che se l'editor e l'assembler devono lavorare simultaneamente, bisognerà comprare una RAM in più.

Il costo di un sistema MDS80 si aggira sui 4.000 dollari ed è poco in confronto a quello delle periferiche che gli vanno aggiunte per renderlo veramente efficace: il costo finale si aggirerà sui 10 - 15.000 dollari.

Quando si valutano i sistemi di sviluppo, molti di essi hanno le attrezzature fondamentali come l'editor, l'assembler ed un debugger, però anche un *conditional macros* può essere considerato molto importante e non è disponibile su tutti i sistemi e la stessa cosa accade per il compilatore locale, quindi, la valutazione dovrebbe essere fatta non tanto in termini di costo complessivo, quanto in termini di disponibilità software.

Presentiamo, infine, uno degli strumenti software più importanti per i sistemi di debugging: l'*emulatore in circuito* (ICE).

Fig. 9-8: In-Circuit Emulator (ICE).

Un *ICE* è un programma che simula l'esecuzione del microprocessore in un tempo quasi reale, esso lo emula. La Fig. 9-8 ne rappresenta una sua versione della Intel. L'ICE risiede all'interno dello stesso mobiletto di sviluppo del sistema e richiede una scheda ulteriore più un cavo esterno che esce dal retro del MDS ed è chiamato il «cordone ombelicale». Esso termina con un morsetto da 40 pin, identico alla piedinatura (pin - out) di un 8080.

Sul sistema utente sotto sviluppo, che appare sulla destra dell'illustrazione, è stata estratta la scheda MPU ed è stata collegata ad un filo di prolunga; il vero 8080 è stato tolto e si è collegato il morsetto a 40 pins all'estremità del cordone ombelicale: l'ICE si comporta esattamente come farebbe un 8080, generando gli stessi se-

gnali sugli stessi piedini.

Qual'è il suo vantaggio? Un vero 8080 non può essere fermato facilmente, non c'è modo di controllare i suoi registri o cambiarli senza un hardware esterno. Sostituendo un programma software al vero 8080, si rende possibile l'uso di tutte le caratteristiche di un potente debugger ed altri programmi di servizio, per controllare l'esecuzione del sistema sotto sviluppo. Si possono fissare dei *breakpoints* (punti d'arresto) e, perciò, il programma può essere eseguito sul sistema vero sulla destra e, quindi, arrestato in corrispondenza di determinati breakpoints: è allora possibile esaminare i registri simbolicamente per esaminare la memoria, o cercare anche delle alternative. I comandi possono essere dati premendo i tasti della tastiera piuttosto che inserire una nuova PROM. Dalla tastiera sarà possibile chiudere valvole, verificare i valori di sensori reali collegati alle vere applicazioni. In più, parte del programma può essere memorizzato nel sistema di sviluppo e, quando le sezioni del programma sono debugged correttamente, possono essere progressivamente impresse nella PROM e quindi inserite nel sistema reale.

Per la prima volta, un ICE offre l'opportunità di verificare ed eseguire il debugging su un sistema reale collegato ai suoi veri dispositivi di input/output in un tempo reale; è questa una delle apparecchiature più potenti che sia stata ideata per il vero debugging di un sistema hardware/software completo. Esso è indispensabile per un debugging efficiente su qualsiasi sistema in tempo reale: nel caso del MDS80, l'emulatore esegue ad una velocità inferiore di solo il 10% rispetto ad un 8080. Ciò è reso possibile dall'utilizzazione di una versione più veloce dell'8080, l'8080 - 2 che opera sulla scheda dell'ICE con un clock di 3 MHz.

Un'altra importante funzione dell'ICE è la possibilità di traccia che andiamo ora a descrivere.

La possibilità di traccia

L'ICE, come molti altri sistemi simili, offre una possibilità di registrazione, chiamata *traccia*. Automaticamente, esso registrerà gli eventi durante i 44 precedenti cicli di macchina prima del suo arresto; è analogo ad un film di eventi all'interno dei 44 cicli precedenti del sistema. Questa rappresenta una funzione fondamentale. Ogni volta che si diagnostica un errore ad un breakpoint è, di solito, troppo tardi; in altre parole, l'errore è stato causato da un'istruzione precedente: il problema essenziale è di identificare l'istruzione che ha causato l'errore ed il punto dove esso è stato diagnosticato. In un programma esistono un certo numero di punti di diramazione ed è spesso molto difficile determinare quale ramo sia stato eseguito prima dell'individuazione dell'errore: è, quindi, essenziale registrare quale cammino nel programma è stato seguito fino al breakpoint. A questo punto, o si può individuare l'istruzione errata o si fisserà un breakpoint anteriore: si registreranno così 44 nuovi cicli della macchina e così via, fino a che l'istruzione errata può essere rintracciata. In aggiunta, è spesso possibile impiegare un video esterno se si necessita di un

debugging fine per un interfaccia hardware complesso.

Le versioni dell'ICE disponibili presso i costruttori sono piuttosto costose (1.200÷1.500 dollari) in confronto al poco hardware impiegato, ma possono essere acquistate anche da rivenditori indipendenti a costi inferiori.

Per concludere, l'ICE è un'attrezzatura essenziale da tener in considerazione se si vuole integrare in breve tempo il debugging hardware/software.

Le alternative

Alcuni venditori indipendenti hanno sviluppato un certo numero di sistemi di sviluppo che forniscono le stesse opportunità e siccome non costruiscono i microprocessori essi stessi, hanno reso i loro sistemi indipendenti da un dato microprocessore, cioé li si può usare per un 8080, un 6800, un 2650 ed altri, facendone dei sistemi ad uso molteplice. In questo modo, essi possono essere particolarmente interessanti per un utente che è restio ad investire troppo denaro in un sistema prima di affidarsi ad un determinato chip, tuttavia, una volta che l'ha fatto, i sistemi di sviluppo forniti dai costruttori forniscono, normalmente, delle migliori opportunità di software per il microprocessore interessato e la prospettiva di continui miglioramenti.

Altri strumenti di sviluppo

Non descriveremo, in questo paragrafo, altri strumenti per l'*individuazione degli errori*, ma un altro importantissimo strumento che è necessario allo sviluppo di

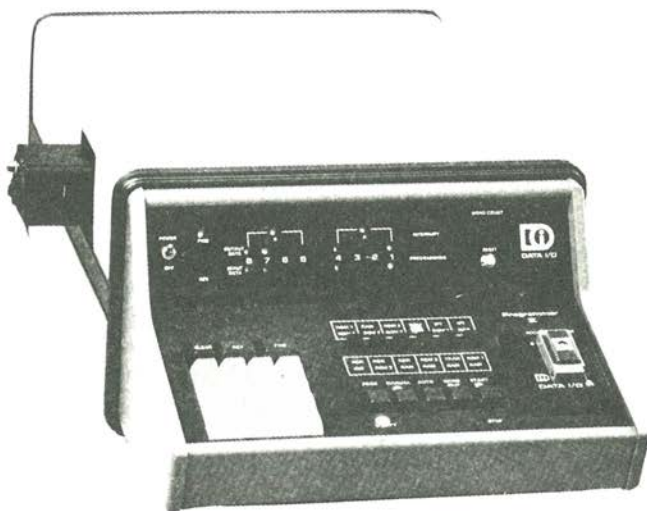



Fig. 9-9: Programmatore PROM (Data I/O).

un'applicazione è il programmatore PROM: il suo compito è di programmare le EPROM o le PROM su cui risiederà il programma. Due esempi sono riportati nelle Figg. 9-9 e 9-10: si tratta di programmatori EPROM per la cancellazione agli ultravioletti (UV) delle PROM. Essi sono muniti di tastiera esadecimale che consente l'immissione manuale dei dati, di un lettore di nastro di carta (o di un collegamento con uno di essi) per consentire al dispositivo di leggere direttamente il programma una volta che esso è stato perforato su nastro. In genere, fornisce anche delle interfacce addizionali quali un RS232 od altri per poterli collegare ad un sistema di microprocessore sul quale risiede il programma. Ci sono, inoltre, altre caratteristiche importanti, quali la duplicazione automatica, la verifica ed altre. Costa normalmente dai 1.000 ai 3.000 dollari.

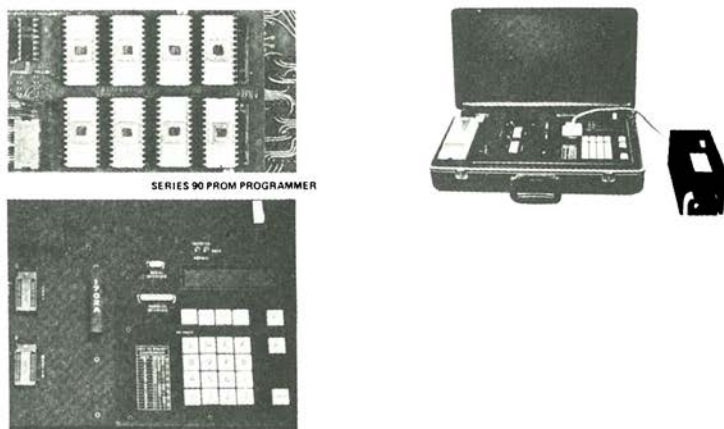



Fig. 9-10: Programmatore PROM (PROLOG).

Numerosi venditori mettono a disposizione un certo numero di *mezzi diagnostici,* in particolare una nuova generazione di strumenti che è stata sviluppata come un'evoluzione dell'analizzatore digitale: l'analizzatore di microprocessore. Sono prodotti, in particolare, dalla Hewlett-Packard, la Tektronix, la Fluke, la Biomation ed altre. Essi sono indispensabili per la sintonizzazione fine ed il debugging di interfacce hardware complesse; il loro costo va dai 5.000 ai 15.000 dollari.

E che dire delle scatole di montaggio (*kits*)? Per uno sviluppo software efficiente non esistono molte altre alternative agli strumenti che abbiamo fin qui presentato, tuttavia, esse esistono se lo scopo è più modesto; esistono, in particolare, i kits o schede assemblate che sono uno strumento ideale per acquistare familiarità con una semplice programmazione a livello di linguaggio di macchina. Normalmente,

non sono equipaggiate con un assembler di modo che la programmazione deve essere eseguita direttamente in esadecimale. I gravami che ne derivano limitano la lunghezza dei programmi che si possono sviluppare su di essi a poche decine o poche centinaia di istruzioni. La memoria di massa che si usa più frequentemente per tale scheda è il comune registratore a cassette che può essere facilmente interfacciato ad essa, conservando perciò i programmi in maniera efficiente. Benché tali schede siano un mezzo didattico molto valido, non possono essere usate da sole come un mezzo di sviluppo vero e proprio per nessun programma reale, tuttavia, esistono molti sistemi modulari che consentono all'utente di inserire delle schede aggiuntive e realizzare progressivamente un vero sistema di sviluppo e ciò rappresenta un'interessante possibilità di evoluzione. Cionostante, come sempre, la spesa maggiore sarà rappresentata non tanto dalla scheda del processore quanto dalle periferiche.

SOMMARIO

Lo sviluppo di un sistema di microprocessore comprende l'uso di tecniche hardware e software che sono state trattate in questo capitolo e nei precedenti. Il problema essenziale al momento dello sviluppo è, di solito, la fase di debugging del software e di questo se ne è parlato ampliamente in questo capitolo e ne sono stati presentati gli strumenti: il più efficiente si è dimostrato il sistema di sviluppo sia per quanto riguarda l'hardware che il software e come alternative abbiamo visto un sistema di time-sharing, un in-house computer ed un kit. Per i sistemi complessi di debugging, specialmente in sistemi a tempo reale, si è visto che uno strumento essenziale è l'ICE. Infine, l'investimento consistente richiesto per un sistema di sviluppo completo si compensa rapidamente in termini di tempi di programmazione ridotti ed un completamento più rapido di un progetto. Tutto questo dovrebbe essere opportunamente preventivato.

Fig. 9-11: Un sistema di sviluppo completo richiede almeno 6 unità (Intel MDS).

CAPITOLO 10

IL FUTURO

INTRODUZIONE

In questo libro abbiamo fatto delle previsioni su come saranno i futuri componenti, sistemi o tecniche e con questo capitolo intendiamo completare l'argomento: predire il futuro può sembrare un'affermazione alquanto azzardata laddove l'evoluzione tecnologica è stata così rapida e la maggior parte delle previsioni tecniche si sono rivelate errate, ma noi dimostreremo che è possibile fare delle ragionevoli previsioni che si rivelino, poi, sicure. Per giustificarle dovremo per prima cosa stabilire il grado di evoluzione della tecnologia, trarne, quindi, le ovvie conclusioni ed alla fine avremo ottenuto una serie di previsioni a breve ed a lungo termine.

Naturalmente, una considerazione essenziale per i costruttori è l'utile di un prodotto; il principale fattore che condiziona il costo dei chips LSI è il rendimento ed esamineremo per primi i parametri che lo influenzano.

IL RENDIMENTO

Il *rendimento* è la percentuale di chips buoni di una determinata partita prodotta ed i problemi che gli sono connessi sono causati da fattori fisici e di lavorazione. A causa delle molteplici difficoltà legate alla produzione di chips quali la contaminazione, le microrotture, un'insufficiente manipolazione accurata od i controlli di processo, i difetti di accoppiamento e delle maschere, un inadeguato allineamento delle maschere e le variazioni di lavorazione, un certo numero di chips risulteranno difettosi.

All'inizio della produzione di un nuovo componente, tutti i chips potrebbero essere difettosi e la lavorazione viene migliorata finché il rendimento cresce progressivamente al punto in cui la produzione diventa remunerativa.

Il parametro essenziale per il rendimento è l'area del chip, come illustrato in Fig. 10-1: il rendimento decresce in maniera esponenziale quando aumenta l'area della chip. In qualsiasi momento dell'evoluzione della tecnologia, ogni costruttore sa come realizzare un chip di un'area data: aumentando l'area moltiplica in maniera esponenziale la probabilità di errore. Al contrario, una volta che un costruttore ha prodotto una quantità sufficiente di chips sarà in grado di raggiungere un rendi-

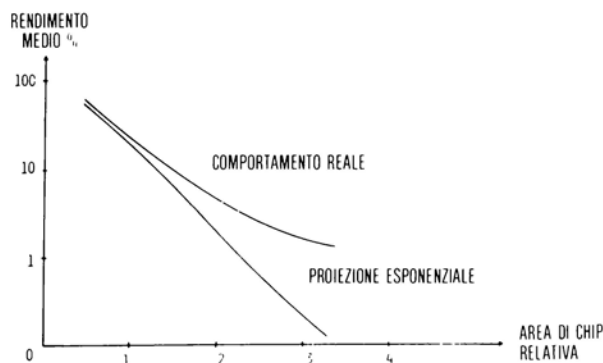



Fig. 10-1: Grafico rendimento - area di chip relativa.

mento maggiore ed anche ridurre l'area del chip come illustrato in Fig. 10-1.

Il risultato di tutto questo è la curva più importante della Silicon Valley: si tratta della cosiddetta *«learning curve»* (Fig. 10-2) la quale stabilisce che: «Il prezzo di un circuito a semiconduttore diminuisce del 30% ogni volta che la produzione globale raddoppia».

Più componenti sono prodotti, meglio si può controllare e migliorare la lavorazione, col risultato di un aumento del rendimento. Come si vede nel grafico di Fig. 10-2 ogni volta che si aumenta il numero dei dispositivi prodotti, il rendimento aumenta ed il prezzo diminuisce: è stato questo il primo fattore che ha determinato il rapido calo del costo dei componenti LSI; il secondo, che ora è prevalente, è stata la concorrenza ed in particolare la concorrenza delle *seconde sorgenti.* Ciò ha portato alcuni costruttori di semiconduttori a fissare dei prezzi che non hanno nulla a

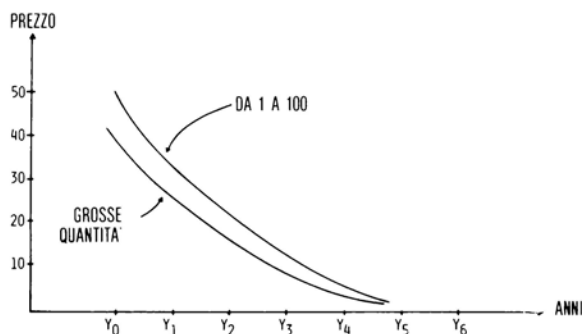



Fig. 10-2: La «learning curve».

che vedere con il loro rendimento; p.e., alcune compagnie particolarmente agguerrite, desiderando introdurre un nuovo chip sul mercato, si saranno accontentate di un rendimento iniziale del 10% però, intanto, dimostravano di poter produrre il chip e, successivamente, speravano che la produzione aumentasse rapidamente fino a raggiungere il 50% od il 90% ed assicurarsi così degli alti profitti. Ecco perché il prezzo dei componenti è strettamente legato al volume prodotto ed ecco perché i microprocessori possono essere venduti in gran quantità per meno di 2 dollari.

L'EVOLUZIONE TECNOLOGICA

Nell'evoluzione tecnica dei prodotti LSI si possono distinguere due tendenze: l'evoluzione verso velocità più elevate e l'evoluzione verso densità più elevate di componenti.

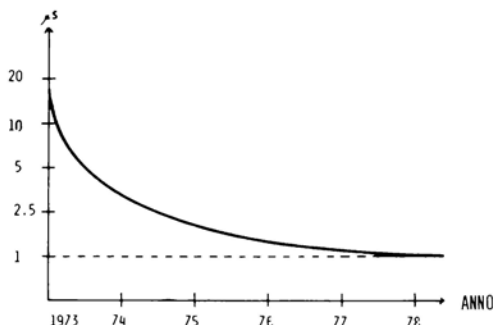



Fig. 10-3: Grafico dell'evoluzione della velocità.

Per *l'evoluzione verso velocità elevate* è stato ora raggiunto un limite; le limitazioni fisiche del processo fotolitografico oggi impiegato limitano il grado di precisione con cui si possono delimitare le varie zone sul silicio, variandone l'efficienza: ne risulta che i limiti attuali nel tempo di esecuzione di istruzioni tipo è dell'ordine di 1 microsecondo. Attualmente, è stato sviluppato un nuovo processo, il raggio elettronico (electron beam), che darà come risultato una velocità più elevata grazie ad una risoluzione più precisa di tali zone. Quando questo entrerà in funzione o quando verranno messi a punto altri processi, questa ricerca di velocità sempre più elevate potrà riprendere.

L'altra importante tendenza è *l'evoluzione verso densità più elevate* di componenti su di un solo chip e questo si può realizzare in due modi: o ridurre il progetto in un'area più ridotta del chip od aumentandone le sue dimensioni: sono questi due

i metodi attualmente seguiti. Le dimensioni dei componenti di un chip sono, oggigiorno, dell'ordine di 10 micron e saranno progressivamente abbassate a 1 micron, che è il limite reale del processo fotolitografico. Inoltre, a mano a mano che si vengono a conoscere sempre meglio i processi, aumenta il rendimento e l'area del chip aumenta regolarmente; non esiste limite teorico all'area massima che può essere utilizzata e non c'è quindi, per ora, nessuna limitazione al numero massimo di componenti per ogni chip.

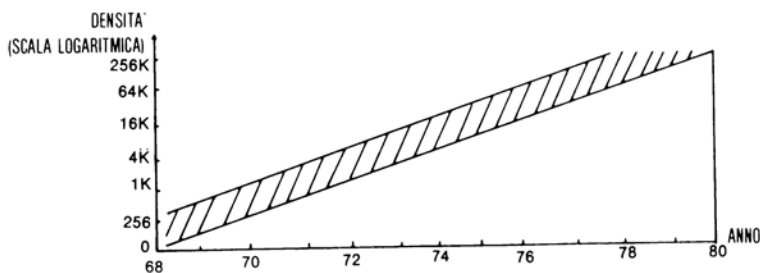



Fig. 10-4: Curva di Moore.

La Fig. 10-4 rappresenta l'evoluzione della densità, misurata dal numero di componenti per chip ed è stata chiamata la «curva di Moore», dal nome del Dr. Moore dell'Intel che per primo la teorizzò: egli afferma semplicemente che la densità delle RAM dinamiche è quadruplicata ogni due anni, cioé raddoppiata ogni anno. L'esperienza fatta nel corso di otto anni sembra aver confermato questa tendenza e si può tranquillamente prevedere che essa continuerà con lo stesso andamento ancora per molti anni. È già possibile ottenere dai 10 ai 15.000 transistors su un chip singolo e stiamo progressivamente andando verso la SLSI (Super-Large-Scale-Integration) con qualcosa come 50.000 transistors su un solo chip. A quel punto, sarà possibile realizzare un computer completo su un solo chip che potrà essere venduto per pochissimi dollari. L'effetto sensazionale di questa affermazione consiste nel fatto che le stesse funzioni che molti anni fa venivano svolte da grossi computer che costavano un milione di dollari o più, verranno svolte, in un futuro ormai prossimo, da componenti di volume ridottissimo e dal prezzo di 1 dollaro; ma il vero, grande impatto non è stato ancora valutato.

Per la prima volta nella storia della nostra era industriale, sarà disponibile una vera forma di «intelligenza» (un'intelligenza automatizzata) di dimensioni e prezzo ridotti. Sarà possibile associarla praticamente a qualsiasi dispositivo fatto dall'uomo, ma è ancora impossibile valutare in pieno le conseguenze di ciò: questo è uno dei motivi per cui l'apparizione dei microprocessori è stata definita «la seconda rivoluzione industriale».

L'EVOLUZIONE DEI COMPONENTI

Cerchiamo ora di trarre le conclusioni sulle previsioni fatte e sull'evoluzione dei componenti che sono state trattate nel nostro libro. L'osservazione di fondo che si può dedurre dall'attuale tecnologia è che l'unità di elaborazione è divenuta una delle risorse più a buon mercato in un sistema e si è reso, quindi, possibile l'impiego dei processori pressoché in qualsiasi applicazione, con risultato che le stesse strutture dei sistemi verranno a modificarsi. Questa «intelligenza», o funzione di elaborazione, verrà associata praticamente a qualsiasi compito del sistema: stiamo entrando in un'era di *elaborazione distribuita*. Nello stesso sistema a microprocessore ciascuno degli input-output e ciascun chip di periferica verranno muniti di elaboratore e si può già prevedere che si venderanno più chips di microcomputer che chips di memoria. La memoria sarà «equipaggiata con il suo microprocessore» (od il microprocessore con la sua memoria) ed allo stesso modo lo saranno il PIO, l'UART, il DAC: ciascuno dei dispositivi LSI diventerà *programmabile*.

In altre parole, i microprocessori saranno presenti ogniqualvolta è presente qualsiasi funzione ed, in particolare, le funzioni input-output. All'ingegnere od al progettista del sistema sarà indispensabile la conoscenza della programmazione per poter progettare un sistema.

L'IMPATTO SOCIALE

Da un punto di vista umano e sociale, ciò significa che i microprocessori diventeranno comuni tanto quanto lo sono i motori elettrici oggigiorno, o forse anche di più. Dozzine di microprocessori saranno presenti nell'ambito della casa e del posto di lavoro. L'introduzione diffusa dei microprocessori porterà, naturalmente, un certo numero di vantaggi; il più significativo è che essi probabilmente soppianteranno una porzione rilevante di forza lavoro; l'automazione della maggior parte dei processi che richiedono dei semplici controlli algoritmici, eliminerà un grosso numero di posti di lavoro. Si apriranno nuove e più stimolanti opportunità.

APPENDICE A

SIMBOLI ELETTRONICI

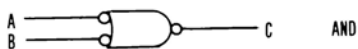
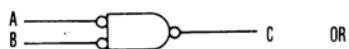
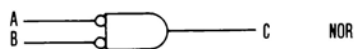
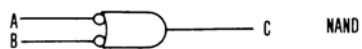
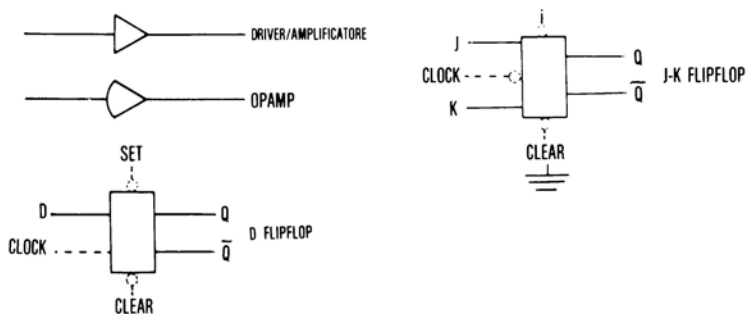
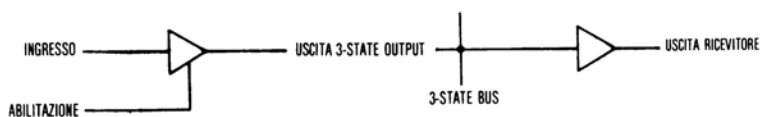
A	$\bar{A}$
0	1
1	0

A	B	A ∧ B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A ∨ B
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

SIMBOLI DI BASE

NEGAZIONI

EQUIVALENZE

ALTRI SIMBOLI

BUFFER TRI-STATE

APPENDICE B

SET DI ISTRUZIONI DEL 6800 DELLA MOTOROLA

ABA	Somma l'accumulatore B all'accumulatore A
ADC	Somma con riporto
ADD	Somma senza riporto
AND	AND logico
ASL	Scorrimento aritmetico a sinistra
ASR	Scorrimento aritmetico a destra
BCC	Diramazione se carry è zero
BCS	Diramazione se carry è uno
BEQ	Diramazione se uguale
BGE	Diramazione se maggiore o uguale a zero
BGT	Diramazione se maggiore di zero
BHI	Diramazione se più alto
BIT	Test del bit (ACCX), (M)
BLE	Diramazione se minore o uguale a zero
BLS	Diramazione se minore o uguale
BLT	Diramazione se minore di zero
BMI	Diramazione se negativo
BNE	Diramazione se non uguale
BPL	Diramazione se positivo
BRA	Diramazione in ogni caso
BSR	Diramazione alla subroutine
BVC	Diramazione se overflow è zero
BVS	Diramazione se overflow è uno
CRA	Confronta gli accumulatori
CLC	Azzera Carry bit C = 0
CLI	Azzera la maschera di interrupt bit I = 0

CLR Azzeramento
ACCX oppure M = 00

CLV Azzera il bit di overflow del complemento a due
bit V = 0

CMP Confronta
(ACCX), (M)

COM Complementa

CPX Confronta con il registro indice
(IX), (M)

DDA Aggiustamento decimale dell'accumulatore A

DEC Decrementa

DES Decrementa il puntatore dello stack
$SP \leftarrow (SP) - 1$

DEX Decrementa il registro indice
$IX \leftarrow (IX) - 1$

EOR OR Esclusivo
$ACCX \leftarrow (ACCX)\ EOR\ (M)$

INC Incrementa

INS Incrementa il puntatore dello stack
$SP \leftarrow (SP) + 1$

INX Incrementa il registro indice
$IX \leftarrow (IX) + 1$

JMP Salto

JSR Salto alla subroutine

LDA Carica l'accumulatore

LDS Carica il puntatore dello stack
$SP \leftarrow (M)$

LDX Carica il registro indice
$IX \leftarrow (M)$

LSR Scorrimento logico a sinistra

NEG Negazione
$ACCX\ oppure\ M \leftarrow \overline{(ACCX\ oppure\ M)}$

NOP Non opera

ORA OR inclusivo

PSH Introduzione dati nello stack

PUL Prelievo dati dallo stack

ROL Rotazione a sinistra

ROR Rotazione a destra

RTI Ritorno da interrupt

RTS	Ritorno da subroutine
SBA	Sottrazione tra gli accumulatori ACCA ← (ACCA) − (ACCB) Sottrazione con riporto
SBC	Sottrazione con riporto ACCX ← (ACCX) − (M) − (C)
SEC	Pone carry ad 1 bit C = 1
SEI	Pone ad 1 la maschera di interrupt bit I = 1
SEV	Pone ad 1 il bit di overflow in complemento a due bit V = 1
STA	Memorizza l'accumulatore M ← (ACCX)
STS	Memorizza il puntatore dello stack
STX	Memorizza il registro indice
SUB	Sottrazione ACCX ← (ACCX) − (M)
SWI	Interrupt software Raccoglie nello stack PC, IX, ACCA, ACCB, CC
TAB	Trasferisce dall'accumulatore A all'accumulatore B ACCA ← (ACCB)
TAP	Trasferisce da ACCA a CC (registro dei codici di condizione) CC ← (ACCA)
TBA	Trasferisce da B ad A ACCA ← (ACCB)
TPA	Trasferisce da CC ad ACCA ACCA ← (CC)
TST	Test
TSX	Trasferisce dal puntatore dello stack al registro indice IX ← (SP) + 0001
TXS	Trasferisce dal registro indice al puntatore dello stack SP ← (IX) − 0001
WAI	Attesa di interrupt
PC	PC + 00001

SIMBOLI

ACCA	Accumulatore A
ACCB	Accumulatore B
ACCX	Accumulatore A oppure B
PC	Contatore di programma
SP	Puntatore dello stack
IX	Registro indice
CC	Registro dei codici di condizione
M	Indirizzo di memoria

APPENDICE C

SET DI ISTRUZIONI DELLA INTEL

( ) = contenuti; r = registro; M = indirizzo di memoria definito da H ed L

MOV r1, r2	Muove r2 nel registro r1 r1 ← (r2): trasferimento di r2 in r1
MOV M, r	Muove un registro nella memoria
MOV r, M	Muove la memoria nel registro r
HLT	Alt
MVI r	Muove immediato il registro r
MVI M	Muove immediato la memoria
INR r	Incrementa il registro r r ← r + 1
DCR r	Decrementa il registro r r ← r − 1
INR M	Incrementa la memoria
DCR M	Decrementa la memoria
ADD r	Somma il registro ad A A ← (A) + (r)
ADC r	Somma il registro ad A con riporto A ← (A) + (r) + C
SUB r	Sottrae il registro da A A ← (A) − (r)
SBB r	Sottrae il registro da A con prestito
ANA r	And del registro con A A ← (A) AND (r)
XRA r	Or esclusivo del registro con A A ← (A) XOR (r)
ORA r	Or del registro con A A ← (A) OR (r)
CMP r	Confronta il registro con A
ADD M	Somma la memoria ad A
ADC M	Somma la memoria ad A con riporto

SUB M	Sottrae la memoria da A
SBB M	Sottrae la memoria da A con prestito
ANA M	And della memoria con A
XRA M	Or esclusivo della memoria con A
ORA M	Or della memoria con A
CMP M	Confronta la memoria con A
ADI	Somma immediato ad A
ACI	Somma immediato ad A con riporto
SUI	Sottrae immediato da A
SBI	Sottrae immediato da A con prestito
ANI	And immediato con A
XRI	Or esclusivo immediato con A
ORI	Or immediato con A
CPI	Confronto immediato con A
RLC	Rotazione a sinistra di A
RRC	Rotazione a destra di A
RAL	Rotazione a sinistra di A attraverso carry
RAR	Rotazione a destra di A attraverso carry
JMP	Salto incondizionato
JC	Salta se c'è riporto
JNC	Salta se non c'è riporto
JZ	Salta se zero
JNZ	Salta se diverso da zero
JP	Salta se positivo
JM	Salta se negativo
JPE	Salta se la parità è pari
JPO	Salta se la parità è dispari
CALL	Chiamata incondizionale
CC	Chiama se c'è riporto
CNC	Chiama se non c'è riporto
CZ	Chiama se zero
CNZ	Chiama se non zero
CP	Chiama se positivo
CM	Chiama se negativo
CPE	Chiama se la parità è pari
CPO	Chiama se la parità è dispari
RET	Ritorno
RC	Ritorno se c'è riporto

RNC	Ritorno se non c'è riporto
RZ	Ritorno se zero
RNZ	Ritorno se diverso da zero
RP	Ritorno se positivo
RM	Ritorno se negativo
RPE	Ritorno se la parità è pari
RPO	Ritorno se la parità è dispari
RST	Riparte
IN	Ingresso
OUT	Uscita
LXI B	Carica immediato la coppia di registri B & C
LXI D	Carica immediato la coppia di registri D & E
LXI H	Carica immediato la coppia di registri H & L
LXI SP	Carica immediato il puntatore dello stack
PUSH B	Introduce nello stack la coppia di registri B & C
PUSH D	Introduce nello stack la coppia di registri D & E
PUSH H	Introduce nello stack la coppia di registri H & L
PUSH PSW	Introduce nello stack A ed i flags
POP B	Preleva dallo stack la coppia di registri B & C
POP D	Preleva dallo stack la coppia di registri D & E
POP H	Preleva dallo stack la coppia di registri H & L
POP PSW	Preleva dallo stack A ed i flags
STA	Memorizza A in modo diretto
LDA	Carica A in modo diretto
XCHG	Scambia i registri D & E, H & L
XTHL	Scambia la sommità dello stack, H & L
SPHL	H & L al puntatore dello stack
PCHL	H & L al contatore di programma
DAD B	Somma B & C ad H & L
DAD D	Somma D & E ad H & L
DAD H	Somma H & L ad H & L
DAD SP	Somma il puntatore dello stack ad H & L
STAX B	Memorizza A in modo indiretto
STAX D	Memorizza A in modo indiretto
LDAX B	Carica A in modo indiretto
LDAX D	Carica A in modo indiretto
INX B	Incrementa i registri B & C
INX D	Incrementa i registri D & E

INX H	Incrementa i registri H & L
INX SP	Incrementa il puntatore dello stack
DCX B	Decrementa B & C
DCX D	Decrementa D & E
DCX H	Decrementa H & L
DCX SP	Decrementa il puntatore dello stack
CMA	Complementa A
STC	Pone carry uguale ad 1
CMC	Complementa carry
DAA	Aggiustamento decimale di A
SHLD	Memorizza in modo diretto H & L
LHLD	Carica in modo diretto H & L
EI	Abilita interrupts
DI	Disabilita interrupt
NOP	Non opera

APPENDICE D

IL BUS S-100 (ALTAIR)

NUMERO DI PIN	SIMBOLO	NOME	FUNZIONE
1	+8V	+8 Volts	Tensione non regolata sul bus, fornita alla piastra PC e regolata a 5 volt.
2	+18V	+18 Volts	Tensione positiva pre-regolata.
3	XRDY	Esterno pronto	Ingresso di esterno pronto verso la circuiteria pronta della piastra CPU.
4	VI0	Interruzione mediante vettore - via 0	
5	VI1	Interruzione mediante vettore - via 1	
6	VI2	Interruzione mediante vettore - via 2	
7	VI3	Interruzione mediante vettore - via 3	
8	VI4	Interruzione mediante vettore - via 4	
9	VI5	Interruzione mediante vettore - via 5	
10	VI6	Interruzione mediante vettore - via 6	
11	VI7	Interruzione mediante vettore - via 7	
12	*XRDY2	Esterno pronto 2	Una seconda via di esterno pronto simile a XRDY.
		* Nuovo segnale del bus per l'8080b	
13 a 17	Da definire		
18	$\overline{\text{STAT DSB}}$	$\overline{\text{Status Disable}}$	Permette che i tamponi per le 8 vie di stato siano tri-state.
19	$\overline{\text{C/C DSB}}$	$\overline{\text{Command/control disable}}$	Permette che i tamponi per le 6 vie di controllo/comando dello stato siano tri - state.
20	UNPROT	Unprotect	Ingresso al flip - flop di protezione memoria nella data piastra di memoria.

NUMERO DI PIN	SIMBOLO	NOME	FUNZIONE
21	SS	Single step	Indica che la macchina è nel processo di realizzazione di un singolo passo (cioé che il flip-flop SS nel D/C è posizionato).
22	ADD DSB	Address disable	Permette che i tamponi per le 16 vie di indirizzo siano a tri-state.
23	DO DSB	Data out disable	Permette che i tamponi per le 8 uscite dati siano tri-state.
24	02	Phase 2 clock	
25	01	Phase 1 clock	
26	PHLDA	Conferma Hold	Segnale di uscita di controllo: comando del processore che appare in risposta al segnale Hold; indica che il bus dati e di indirizzo passerà allo stato HOLD dopo il completamento del ciclo di macchina corrente
27	PWAIT	Attesa	Segnale di controllo/comando del processore che appare in risposta al segnale READY che sta passando a livello basso; indica che il processore entrerà in una serie di stati di attesa di 0,5 μsec finché READY diventa nuovamente alto.
28	PINTE	Interrupt enable	Segnale di uscita di controllo/comando del processore; indica che le interruzioni sono abilitate, come determinato dai contenuti del flip-flop di interruzione interno alla CPU. Quando il flip-flop è posto a set (istruzione abilitazione interruzione) le interruzioni sono accettate dalla CPU; quando esso è resettato (istruzione disabilitazione interruzione) le interruzioni sono inibite.
29	A5	Via di indirizzo 5	
30	A4	Via di indirizzo 4	
31	A3	Via di indirizzo 3	
32	A15	Via di indirizzo 15	(MSB)
33	A12	Via di indirizzo 12	
34	A9	Via di indirizzo 9	
35	DO1	Via dati in uscita 1	
36	DO0	Via dati in uscita 0	(LSB)
37	A10	Via dati in uscita 10	
38	DO4	Via dati in uscita 4	

NUMERO DI PIN	SIMBOLO	NOME	FUNZIONE
39	DO5	Via dati in uscita 5	
40	DO6	Via dati in uscita 6	
41	DI2	Via dati in ingresso 2	
42	DI3	Via dati in ingresso 3	
43	DI7	Via dati in ingresso 7	(MSB)
44	SM1	Machine Cycle 1	Segnale di uscita di stato che indica che il processore è nella fase di prelievo del primo byte di una istruzione
45	SOUT	Output	Segnale di uscita di stato che indica che il bus di indirizzo contiene l'indirizzo di un componente in uscita e che il bus dati conterrà i dati in uscita quando PWR è attivo
46	SINP	Input	Segnale di uscita di stato che indica che il bus di indirizzo contiene l'indirizzo di un componente di ingresso e che dati in ingresso potranno essere posti nel bus dati quando PDBIN è attivo
47	SMEMR	Memory read	Segnale di uscita di stato che indica che il bus dati sarà usato per la lettura di dati da memoria
48	SHLTA	Halt	Segnale di uscita di stato che conferma una istruzione di HALT
49	CLOCK	Clock	Uscita invertita di 02 CLOCK
50	GND	Ground	
51	+8V	+8 Volts	Ingresso non regolato al regolatore a 5 volt
52	-18V	-18 Volts	Tensione negativa pre-regolata
53	SSWI	Sense switch input	Indica che sta per aver luogo un trasferimento dagli switch di acquisizione. Tale segnale è usato dalla logica di schermo. Controllo per: a) abilitare i pilotaggi degli switch acquis.;
		b) abilitare i pilotaggi di ingresso dati della piastra di controllo/visualizz. (FD10-FD17);	c) disabilitare i pilotaggi di CPU di ingresso dati (D10-D17).

Lista parziale dei segnali. Il listing completo è riportato nel libro: «Tecniche di interfacciamento dei microprocessori».

APPENDICE E

COSTRUTTORI

AMD (Advanced Micro Devices)
901 Thompson Place
Sunnyvale, CA 94608
(408) 732-2400
Telex: 346306

AMI (American Microsystems)
3800 Homestead Road
Santa Clara, CA 95051
(408) 246-0330

DATA GENERAL
Southboro, MASS 01772
(617) 485-9100
Telex: 48460

ELECTRONIC ARRAYS
550 East Middlefield Road
Mountain View, CA 94043
(415) 964-4321

FAIRCHILD SEMICONDUCTOR
1725 Technology Drive
San Jose, CA 95110
(408) 998-0123

GI (General Instruments)
600 West John Street
Hicksville, NY 16002
(516) 733-3107
TWX: (510) 221-1666

HARRIS SEMICONDUCTOR
Box 883
Melbourne, FLA 32901
(305) 724-7430
TWX: (510) 959-6259

INTEL
3065 Bowers Avenue
Santa Clara, CA 95051
(408) 246-7501
Telex: 346372

INTERSIL
10090 North Tantau Avenue
Cuppertino, CA 95014
(408) 996-5000
TWX: (916) 338-0228

MMI (Monolithic Memories)
1165 East Arques Avenue
Sunnyvale, CA 94086
(408) 739-3535

MOS TECHNOLOGY
950 Rittenhouse Road
Norristown, PA 19401
(215) 666-7950
TWX: (510) 660-4033

MOSTEK
1215 West Crosby Road
Carollton, TX 75006
(214) 242-0444
Telex: 30423

MOTOROLA SEMICONDUCTOR
Box 20912
Phoenix, ARIZ 85036
(602) 244-6900
Telex: 67325

NS (National Semiconductor)
2900 Semiconductor Drive
Santa Clara, CA 95051
(408) 732-5000
TWX: (910) 339-9240

RAYTHEON SEMICONDUCTOR
350 Ellis Street
Mountain View, CA 94042
(415) 968-9211
TWX: (910) 379-6481

RCA SOLID STATE
Box 3200
Somerville, NJ 08876
(201) 722-3200
TWX: (718) 480-9333

ROCKWELL INTERNATIONAL
Box 3669
Anaheim, CA 92803
(714) 632-3698

SIGNETICS
811 East Arques Avenue
Sunnyvale, CA 94086
(408) 739-7700

SYNERTEK
3050 Coronado Drive
Santa Clara, CA 95051
(408) 984-8900
TWX: (910) 338-0135

TI (Texas Instruments)
Digital Systems Division
P.O. Box 1444
Houston, TX 77001
(713) 494-5115

WESTERN DIGITAL CORP.
3128 Redhill Avenue
Newport Beach, CA 92663
(714) 557-3550
TWX: (910) 595-1139

ZILOG
170 State Street
Los Altos, CA 94022
(415) 526-2748
TWX: (910) 370-7955

APPENDICE F

ACRONIMI

AC	Alternating Current *Corrente alternata*
ACC	Accumulator *Accumulatore*
ACK	Acknowledge *Conferma ricezione*
A/D	Analog to Digital *Da analogico a digitale*
ADCCP	Advanced Data Communication Control Procedure *Procedura di controllo di comunicazione dati avanzata*
ALU	Arithmetic-Logic Unit *Unità aritmetico-logica*
ANSI	American National Standards Institute *American National Standard Institute*
ASCII	American Standard Code for Information Interchange *American Standard Code for Information Interchange*
ASR	Automatic Send and Receive *Trasmissione-ricezione automatiche*
BCD	Binary-Coded-Decimal *Decimale codificato in binario*
BCR	Byte Count Register *Registro conteggio di byte*
BPS	Bits Per Second *Bit per secondo*
BRA	Branch, go to *Ramificazione (branch), «go to»*

BSC Binary Synchronous Communication
Comunicazione sincrona binaria

C Carry
Riporto

CAD Computer-Aided-Design
Progetto con l'ausilio del calcolatore

CAM Contents-Addressable Memory
Memoria indirizzabile su contenuto

CCD Charge-Coupled Device
Dispositivo ad accoppiamento di carica

CE Chip Enable
Abilitazione di modulo

CLK Clock
«Clock» – Temporizzatore

CML Current Mode Logic
Logica di modo corrente

CMOS Complementary MOS
MOS complementare

CPG Clock Pulse Generator
Generatore di impulsi di clock

CPS Characters Per Second
Caratteri al secondo

CPU Central Processor Unit
Unità Centrale di elaborazione

CR Card Reader; Carriage Return
Lettore di Carta – Ritorno carrello

CRC Cyclic Redundancy Check
Controllo di ridondanza ciclica

CROM Control-ROM
ROM di controllo

CRT Cathode Ray Tube
Tubo a raggi catodici

CRTC CRT Controller
Controllore di CRT

CS Chip Select
Selettore di modulo

CTS Clear to Send
«Clear to send»

CU Control Unit
Unità di controllo

CY Carry
Riporto

D Data
Dati

D/A Digital to Analog
Da digitale ad analogico

DC Direct Current
Corrente continua

DC Don't Care
Qualsiasi valore («don't care»)

DCD Data Carrier Detect
Rivelatore di portante dati

DIP Dual In-Line Package
Incapsulamento «Dual In-Line»

DMA Direct Memory Access
Accesso diretto in memoria

DMAC DMA Controller
Controllore DMA

DMOS Double-Diffused MOS
MOS a doppia diffusione

DNC Direct Numerical Control
Controllo numerico diretto

DOS Disk Operating System
Sistema operativo basato su disco

DPM Digital Panel Meter
Misura su pannello digitale

DTL Diode-Transistor Logic
Logica basata su diodi-transistori

DTR Data Terminal Ready
«Data terminal ready»

D0-7 Data Lines 0 Through 7
Vie dati da 0 a 7

E Empty; Enable (Clock)
Vuoto; Abilitazione (Clock)

EAROM Electrically Alterable ROM
ROM alterabile elettricamente

EBCDIC Extended Binary - Coded-Decimal Information Code
Codice d'informazione digitale codificato in binario esteso

ECL Emitter Coupled Logic
Logica di accoppiamento di e-mettitore

EDP Electronic Data Processing
Elaborazione dati elettronica

EFL Emitter Follower Logic
Logica ad inseguimento di e-mettitore

EMI Electro Magnetic Interference
Interfaccia elettromagnetica

EOC End of Conversion
Fine della conversione

EOF End of File
Fine del file

EOR Exclusive OR
OR esclusivo

EOT End of Text, Tape
Fine della trasmissione

EPROM Erasable PROM
PROM cancellabile

FAMOS Floating-Gate Avalanche MOS
Porta fluttuante a valanga MOS

FDC Floppy-Disk Controller
Controllore di disco floppy

FDM Frequency-Division Multiplexing
Multiplazione a divisione di frequenza

FET Field-Effect Transistor
Transistore ad effetto di campo

FF Flip-Flop
Flip-flop

FIFO First-In-First-Out
Primo in ingresso-primo in u-scita

FPLA Field PLA
PLA mediante campo

FSK Frequency-Shift-Keying
Chiave a scorrimento di frequenza

G (carry) Generate
Generazione (di carry)

GP General-Purpose
Per applicazioni generali

GPIB General-Purpose Interface Bus
Bus di interfaccia GP

HDLC High Level Data Link Control
Accoppiamento per dati su livello alto

HEX Exadecimal
Esadecimale

HPIB Hewlett-Packard Interface Bus
Bus di interfaccia Hewlett-Packard

I Interrupt/Interrupt Mask
Interruzione/maschera di interruzione

IC Integrated Circuit = Chip
Circuito integrato (Chip)

INT Interrupt
Interruzione

I/O Input/Output
Ingresso/uscita

IOCS I/O Control System
Sistema di controllo I/O

IRQ Interrupt Request
Richiesta di interruzione

I^2L Integrated Injection Logic
Logica di integrazione ad iniezione di corrente

JAN Joint Army-Navy
Joint Army-Navy

JP Jump
Salto

K (1024) Kilo
Kilo (1024)

KSR Keyboard-Send-Receive
Tastiera di trasmissione/ricezione

LCD Liquid-Crystal Display
Visualizzazione a cristalli liquidi

LED Light Emitting Diode
Diodo ad emissione di luce

LIFO Last-In-First-Out
Ultimo entrato-primo in uscita

LOC Loop On-Line Control
Controllo di anello (loop) «on line»

LP Line Printer
Stampante per linee

LPM Lines Per Minute
Linee al minuto

LPS Low-Power Shottky
Low-Power Shottky

LRC Longitudinal Redundancy Check
Controllo a ridondanza longitudinale

LSB Least Significant Bit
Bit meno significativo

LSI Large Scale Integration
Integrazione a larga scala

MNOS Metal Nitride Oxide Semiconductor
Metallo-Nitruro-Ossido-Semiconduttore

MOS Metal Oxide Semiconductor
Metallo-Ossido-Semiconduttore

MPU Microprocessor Unit
Unità microprocessore

MSB Most Significant Bit
Bit più significativo

MSI Medium Scale Integration
Integrazione a media scala

MTBF Mean Time Between Failures
Intervallo medio tra guasti successivi

MUX Multiplexer
Multiplatore

N Negative (Sign Bit)
(bit di segno) negativo

NDRO Non-Destructive Read-Out
Lettura non distruttiva

NMOS N-Channel MOS
MOS a canale N

NVM Non-Volatile Memory
Memoria non volatile

OCR Optical Character Reader
Lettore ottico di caratteri

OEM Original Equipment Manufacturer
Costruttore di apparecchiatura originale

OP Operation
Operazione

OV Overflow
Overflow

P Parity; (carry) Propagate
Parità; propagazione (del carry)

PABX Private Automatic Branch Exchange
Commutatore automatico privato di agenzia

PBX Private Branch Exchange
Commutatore privato di agenzia

PC Printed Circuit; Program Counter
Circuito stampato; Contatore di programma

PCI/O Program Controlled I/O
I/O controllato da programma

PCM Pulse Code Mod.
Modulazione a codifica di impulso

PFR	Power-Fail Restart *Ripristino per caduta di alimentazione*
PIC	Priority Interrupt Control *Controllo prioritario di interruzioni*
PIO	Programmable I/O Chip/Interface *Circuito di interfaccia programmabile di I/O*
PIT	Programmable Interval-Timer *Temporizzatore programmabile di intervallo*
PLA	Programmable Logic-Array *Trama logica programmabile*
PLL	Phase-Locked Loop *Anello ad aggancio di fase*
PMOS	P-Channel MOS *MOS a canale P*
POS	Point-of-Sale Terminal *Terminale per punto di vendita*
PROM	(Field) Programmable ROM *ROM programmabile in «campo»*
PSW	Program Status Word *Parola di stato di programma*
PTP	Paper Tape Punch *Nastro di carta perforato*
PTR	Paper Tape Reader *Lettore di nastro di carta*
Q	AC extension *Estensione di AC*
QPL	Qualified Products List *Lista di prodotti omologati*
R	Read *Lettura*
RALU	Register Arithmetic Logic Unit *Registro di unità logico-aritmetica*
RAM	Random-Access-Memory *Memoria ad accesso casuale*
RDSR	Receiver Data Service Request *Richiesta di servizio dal ricevitore dati*
RDY	Ready *Pronto*
RES	Reset *«reset»*
RF	Radio Frequent *Radiofrequenza*
RMS	Root Mean Square *Valore quadratico medio*
ROM	Read-Only Memory *Memoria a sola lettura*
RPROM	Reprogrammable PROM *PROM programmabile*
RPT	Repeat *Ripeti*
RS	Register Select *Selezione di registro*
RST	Restart *Ri-inizializzazione*
RTC	Real-Time Clock *Clock «real-time»*
RTS	Request-To-Send *«Request-to-send»*
R/W	Read/Write Memory *Memoria di lettura/scrittura*
Rx	Receiver *Ricevitore*
SAR	Successive Approximation Register *Registro per approssimazioni successive*
SDLC	Synchronous Data Link Control *Controllo di giunzione dati sincrona*
SEC	Scanning Electron Microscope *Microscopio a scansione elettronica*
SEM	Standard Electronic Module *Modulo di elettronica standard*

S/H	Sample and Hold
Campionamento e mantenimento

S/N	Signal to Noise
Segnale rispetto al rumore

SOS	Silicon-On-Sapphire
Silicio su zaffiro

SR	Service Request
Richiesta servizio

SSI	Small Scale Integration
Integrazione su piccola scala

STB	Strobe
Finestra di abilitazione (strobe)

SUB	Subroutine
Subroutine (sottoprogramma)

TDM	Time-Division Multiplexing
Multiplazione a divisione di tempo

TDSR	Transmitter Data Service Request
Richiesta di servizio dati da trasmettitore

TSS	Time-Sharing System
Sistema a divisione di tempo

TTL	Transistor Transistor Logic
Logica basata su Transistori-transistori

TTY	Teletypewriter
Telescrivente

Tx	Transmitter
Trasmettitore

UART	Universal Asynchronous Receiver Transmitter
Trasmettitore-ricevitore universale asincrono

μC	Microcomputer
Microcalcolatore

μP	Microprocessor
Microprocessore

USRT	Universal Synchronous Receiver Transmitter
Trasmettitore-ricevitore universale sincrono

U-V	Ultra-Violet
Ultravioletto

VMOS	Vertical MOS
MOS verticale

V_{ss}	Ground
Massa

W	Write
Scrittura

WPM	Words Per Minute
Parole al minuto

X	Index
Indice

XOR	Exclusive OR
OR esclusivo

Z	Zero Bit
Bit zero

Φ	(Clock) Phase
Fase (di clock)

Edizione Italiana del
MICROPROCESSOR INTERFACING TECNIQUES

L. 25.000

della

Inc.

Cod. 320P

GRUPPO
EDITORIALE
JACKSON

RODNAY ZAKS

dai chips ai sistemi
microprocessori

29